Développement d'outils informatiques appliqués aux biotechnologies

Alban Gaignard
alban.gaignard@ifsic.univ-rennes1.fr
alban@newbiotechnic.com

sous la direction de
Francisco Javier González

Rapport de stage de fin d'études, en vue de l'obtention
du diplôme d'Ingénieur de l'IFSIC
Diic filière *Langages et Systèmes Informatiques* (LSI)

Newbiotechnic, S.A.

http ://www.newbiotechnic.com
Paseo de Bollullos de Mitacion, 6
Parque Industrial A-49 (PIBO)
41110 Bollullos de Mitacion

Sevilla - España

Université de Rennes1, IFSIC
Année 2003-2004

Remerciements

Je tiens à remercier en premier lieu l'ensemble du personnel de NEWBIOTECHNIC pour leur acceuil, et le climat d'écoute et d'échange auquel tout le monde a contribué, afin de rendre le travail agréable et efficace.

Ils sont tout spécialement destinés à mon maître de stage FRANCISCO JAVIER GONZÁLES pour son soutien tout au long de ce séjour, sa patience, ses remarques précises, et la façon avec laquelle il m'a guidé vers les compromis les plus acceptables.

Je tiens également à remercier MANUEL REY BARRERA docteur en biologie et responsable du département de biologie moléculaire, pour ses éclaircissements sur les notions biologiques que je ne maitrisais pas. Ces remerciements s'adressent aussi à RAFAEL CAMACHO FUMANAL, directeur de NEWBIOTECHNIC, pour les facilités qu'il a pu me fournir, notamment en m'offrant à mon arrivée un logement dans une résidence me laissant le temps de trouver un hébergement fixe sur Séville.

Enfin je souhaite remercier particulièrement JACQUES NICOLAS, responsable du projet *Symbiose* de l'IRISA pour le temps qu'il a pu me consacrer avant mon départ, afin de m'expliquer les activités informatiques liées aux projets de séquençage d'ESTs, et MARIE-THÉRÈSE MISSET, professeur de l'université de Rennes1, pour ses réponses à mes questions d'ordre biologique.

Résumé

Les avancées de ces quinze dernières années en génétique, caractérisées par exemple, par le décryptage de génomes entiers, sont à l'origine de la naissance et de la croissance d'un grand nombre d'entreprises dédiées aux biotechnologies. En effet, ces gènes décryptés sont déterminants des caractéristiques et fonctionalités biologiques du vivant.

Les données générées et devant être manipulées par ces activités sont devenues tellement massives qu'il n'est plus maintenant conçevable de travailler sans l'aide de l'informatique. C'est en effet devenu dans ces domaines un outil fondamental de fouille, gestion, et traitement de données pour la détermination, l'identification et l'analyse fonctionnelle, des séquences génomiques, transformant l'Information en Savoir. Celles qui ont répondu à certains critères déterminés à l'aide d'algorithmes (traitement "in silico"), peuvent faire l'objet d'expérimentations coûteuses "in vivo" afin de prouver leur responsabilité dans un mécanisme biologique. Elle a également une importance notoire dans tous les problèmes d'évaluation, et de traitement statistiques de résultats d'expérimentation. Ce présent document synthétise les activités que j'ai effectuées durant mon séjour chez NEWBIOTECHNIC, du début du mois de mars 2004 à la fin du mois d'août 2004, dans les domaines de l'analyse d'image, de la maintenance et la mise à jour d'une application Web, et la conception suivie de la mise en oeuvre d'un système réparti de comparaison et d'annotation de séquences génomiques.

L'analyse d'image a été mise en oeuvre dans un problème de segmentation d'images bi-modales, destiné à quantifier la croissance des racines au cours d'expèrimentation. L'application se base sur le logiciel java *opensource* de traitement d'images scientifiques IMAGEJ, et se présente sous la forme d'une interface utilisateur affichant l'image originale et les résultats de la segmentation (contours et table de mesure de surface de chacune des entités).

Le travail sur l'application Web s'est déroulé en deux étapes, la réalisation de graphes de documentation de cette application à architecture trois-tiers dans un premier temps, et le développement de nouvelles fonctionnalités liées à la mise à jour d'une base de données météorologique.Il a été accompagné par diverses modifications d'interface utilisateur, et d'extraction d'informations sur les erreurs survenues.

Le développement d'une plateforme de comparaison réparti de séquences génomique se fonde quant à lui sur le *grid computing* et a consisté en la prise en main de la technologie de grille de calcul fournie par GRIDSYSTEMS, du développement de modules pour celle-ci, et enfin du développement d'une plateforme d'analyse fonctionnelle de séquences, pilotée par une interface homme-machine. Elle permet la recherche des séquences dans une banque de donnée, la création d'un module BLAST pour la grille, et le traitement des résultats de comparaison afin de les stocker également dans une base de données.

Ce document s'articulera donc, après une courte présentation de l'entreprise, autour de ces trois thèmes qui seront plus précisément introduits (notamment resitués dans leur contexte), dans chacun des trois chapitres avec un ordre relatif à la chronologie de ces travaux.

Abstract

Breakthroughs on Genetics since the last 15 years, fueled by the cracking of whole genomes, are the base for the start-up and growth of a great number of Biotech companies. Indeed, these deciphered genes determine the main characteristics and biological functions of every living organism.

The amount of raw data produced and to be handled by these research activities is extremely huge and completely unworkable without the assistance of data processing techniques. Computing has become an essential tool for collection, management and exploitation of data to determine, identify and elucidate the function of genomic sequences, turning information into knowledge. Massive "in silico" screening of sequences is made by algorithms, and only those matching selected criteria will be tested by expensive "in vivo" experiments to prove their putative biological function. Computer science is also the cornerstone to resolve problems of evaluation and statistical processing of experimental results. This report summarizes the activities carried out during my training period at NEWBIOTECHNIC, S.A. from March 2004 to August 2004, in the fields of image analysis, maintenance and update of a Web application, and design and deployment of a parallel computing framework for the comparison and annotation of genomic sequences.

The image analysis was implemented to solve a problem related to bimodal images segmentation, aimed to quantify the growth of plant roots during a experimetal period. The application is based on the java opensource software for scientific image processing ImageJ, and looks like a graphic user interface displaying side-by-side the original image and the result of the segmentation (outlines and measured area of each entity).

The work on the Web application had two steps: the graphical documentation of this three-tier architecture application was firstly developed, then new features on the meteorological database updating process were included. Several improvements on the user interface, as well as data collection for bug tracking, were also implemented.

The development of the genomic sequence comparison framework is based on a grid computing scheme. This tailor-made solution, based on the commercial grid technology provided by GRIDSYSTEMS, includes additional modules programming and development of a platform for functional analysis of DNA sequences driven by a graphical user interface. It allows searching of sequences in a database, the creation of a BLAST module for the grid, the processing of comparison results, and finally to keep them stored in a database.

This report will thus focus - after a short presentation of the company - on these three topics which will be further introduced by background explanations within their respective scenarios, and developed in each one of the chapters sorted according to the chronology of the works.

Table des matières

Abréviations **5**

1 Newbiotechnic S.A. (N.B.T.) **6**
1.1 Profil de l'entreprise, mission . . . 6
1.2 Produits et Technologies . . . 6
1.3 Où les Technologies de l'Information interviennent . . . 7
1.4 Besoins en Développements Informatiques . . . 7
1.5 L'équipe . . . 7

2 Mesure semi-automatique de la surface de racines **8**
2.1 Besoins, spécifications informelles . . . 8
2.1.1 Contexte biologique . . . 8
2.1.2 Outils existants pour la réalisation de cette analyse d'images . . . 8
2.2 Expression du problème, et notions théoriques pour le résoudre . . . 10
2.2.1 Objectifs de ce système d'analyse d'images . . . 10
2.2.2 Notions d'analyse d'image . . . 11
2.3 Différentes étapes d'élaboration de ce système . . . 13
2.3.1 Les images . . . 13
2.3.2 Protocole de traitements avec ImageJ . . . 13
2.3.3 Développement d'un *plugin* ImageJ . . . 13
2.3.4 Développement d'une IHM . . . 16
2.4 Mise en perspective, regard critique . . . 18

3 Maintenance et mise à jours d'une application web **20**
3.1 Contexte . . . 20
3.2 Problèmatique . . . 21
3.3 Travail réalisé . . . 21
3.3.1 Documentation et maintenance . . . 21
3.3.2 Ajout de fonctionnalités . . . 22
3.4 Mise en perspectives . . . 22

4 Comparaison répartie de séquences génomiques **24**
4.1 Introduction thématique . . . 24
4.1.1 Quelques notes de Biologie . . . 24
4.1.2 Blast : Alignements et Similarités de séquences . . . 25
4.1.3 *"Grid Computing"* . . . 29

4.2 Problématique . . . 29
4.3 Projets similaires, outils disponibles . . . 29
4.3.1 InnerGrid . . . 29
4.3.2 BioJava . . . 30
4.4 Plateforme logicielle : BlastOnAGrid . . . 30
4.4.1 Pré-traitement des données . . . 32
4.4.2 Traitement des données . . . 33
4.4.3 Post-traitement des données . . . 34
4.4.4 Choix de conception, modélisation . . . 37
4.4.5 Interface Homme-Machine . . . 37
4.4.6 Diffucités techniques rencontrées . . . 38
4.4.7 Evaluation des performances . . . 38
4.5 Mise en perspective . . . 38

5 Prise de recul et conclusions 39
5.1 Organisation . . . 39
5.2 Difficultés d'ordre général rencontrées . . . 39
5.3 Travail réalisé . . . 40
5.4 Conclusion . . . 40

A Mesure de croissance de Racines via ImageJ 42
A.1 Fonctionnement de l'application de mesure . . . 42
A.2 Document diffusé au sein d'NBT, au département "phytopathologie" . . . 43
A.3 Manual processing . . . 43
A.3.1 Image to process : the photo of roots . . . 43
A.3.2 Pre-processing . . . 43
A.3.3 Analysis of particle : counting and measurement . . . 45
A.3.4 Last minute . . . 47
A.4 Automatic processing . . . 47
A.4.1 How To . . . 47
A.4.2 Summary of the process . . . 47
A.4.3 Things that are not yet implemented . . . 48
A.4.4 *To Do* . . . 48

B Gestion Intégrée de l'Oliveraie 49

C BlastOnAGrid 51

Bibliographie 53

Abréviations

ADN acide désoxyribonucléique

ARN acide ribonucléique

BLAST *Basic Local Alignement Search Tool*

EJB *Enterprise Java Bean*

EST *Expressed Sequence Tag*

EstASA *Est semi-Automated Sequence Analysis*

GIO *Gestion Integral del Olivo* gestion intégrée de l'oliveraie

HTML *Hypertext markup language*

IHM Interface homme machine

INM *Instituto Nacional de Meteorología* analogue a *Meteo France*

Jar *Java archive*

JDBC *Java database connectivity*

JDK *Java development kit*

JRE *Java runtime environment*

JSP *Java server pages*

LAN *Local area network*

NBT *NewBioTechnic*

NIMP Norme internationale pour les mesures phytosanitaires [CEMP, 1996]

SQL *Structured Query Language*

UML *Unified modeling language*

WAN *Wide area network*

XML *eXtensible markup language*

XSL *eXtensible stylesheet language*

Chapitre 1

NEWBIOTECHNIC S.A. (N.B.T.)

1.1 Profil de l'entreprise, mission

NEWBIOTECHNIC S.A. est une entreprise privée de Sevilla (España). Fondée en Janvier 1999 par deux chercheurs des universités de Salamanque et Seville, et soutenue financièrement par l'établissement bancaire *EL MONTE Caja de Huelva y Sevilla*, NBT est une entreprise *spin-off*, issue d'une scission par branches d'activités. Par son origine et nature, NBT est une entreprise *knowledge intensive* dédiée principalement, à la recherche scientifique, au développement technologique et à l'innovation. L'équipe, hautement qualifiée, est constituée de 23 salariés effectifs (dont six Docteurs et dix *licenciados*[1]) et des collaborateurs externes : quatre conseillers scientifiques de haut niveau, un directeur commercial et marketing, un conseiller en propriété industrielle et brevets biotechnologiques , et des stagiaires de recherche. De plus, l'entreprise conserve une collaboration avec plus de 45 groupes appartenant à des centres publics de recherche, constituant un réseau de recherche et développement dans lequel figurent plus de 200 scientifiques.

Les activités de NEWBIOTECHNIC sont particulièrement centrées sur les domaines de l'alimentaire et de l'agriculture. NBT est impliquée dans divers projets de recherche et developpement, pour l'élaboration d'outils microbiologiques et moléculaires dans les domaines de l'agronomie : amélioration des rendements de production agricole, applications phytosanitaires, nouvelles alternatives aux outils agro-chimiques, additifs à risques limités pour l'industrie alimentaire... L'objectif principal de ces projets est l'identification de marchés et le transfert de solutions innovatrices dans ces domaines.

1.2 Produits et Technologies

Les produits et services comercialisés par NBT s'organisent en 3 groupes :

Biocontrôle et applications agronomiques produits ayant pour but de favoriser la croissance, le rendement de plantes, un fongicide[2] biologique basé sur le champignon *Trichoderma* (c.f. 2.1.1), un outil (GIO c.f. 3.1) basé sur un portail Web, permettant d'optimiser l'irrigation et la fertilisation des oliviers en Andalousie.

Diagnostics génétiques service de séquençage d'ADN, de bioinformatique, d'intégration et d'analyse *in silico* [3] de données ; diagnostics humains et vétérinaires, détection

[1] diplôme de fin d'études universitaires équivalent au DESS français
[2] produit détruisant les champignons parasites
[3] analyse de données par des algorithmes informatiques

d'organismes génétiquement modifiés, de fraudes pour des produits alimentaires de certaines variétés.

Technologies génétiques gènes destinés à améliorer la résistance et tolérance de plantes à certaines maladies, améliorations de leur production par modification génétique.

1.3 Où les Technologies de l'Information interviennent

Le département de bioinformatique, dont Francisco Javier Gonzáles est le responsable, se charge de mettre en place des infrastructures informatiques ayant pour but de stocker, manipuler et analyser les données issues de projets de séquençage génomiques(chapitre 4). Par ailleurs, outre les besoins liés au fonctionnement de l'entreprise, l'informatique intervient dans un produit/service commercialisé par NBT permettant la gestion intégrée de l'oliveraie (GIO). Il se présente coté utilisateur et administrateur, sous la forme d'une interface web permettant de piloter l'application. Elle fera l'objet du chapitre 3. L'informatique intervient aussi dans tous les problèmes d'analyse de résultats d'expériences biologiques, comme par exemple lors de la quantification de la croissance de plantes afin de déterminer les effets d'additifs nutritifs(par exemple). Cette mesure s'effectue par analyse d'images, et fait l'objet du chapitre 2.

1.4 Besoins en Développements Informatiques

Le sujet de stage n'ayant pas été défini de manière extrêmement précise, mon séjour à NBT m'a permi d'aborder les thèmes précédeİment évoqués, qui sont : l'analyse d'image, la maintenance et l'évolution d'une application Web, et l'analyse et l'annotation de séquences génomiques basées sur une technologie novatrices qui est le *grid computing*. Par ailleurs, j'ai eu à effectuer des tâches d'administration de systèmes dans un réseau hétérogène d'une quinzaine de machines fonctionnant avec Linux, MacOS X et différentes versions de Windows. Il a fallu aussi maintenir deux serveurs dont un Windows et l'autre Linux, et se rendre disponible pour des questions liées à l'utilisation d'outils bureautiques.

1.5 L'équipe

J'ai été dirigé tout au long de ce séjour chez NBT, par Francisco Javier Gonzáles (*Fran*), responsable du département de bioinformatique, docteur en pharmacie, et expert en bioinformatique et calcul intensif (il a en effet été responsable du centre de calcul de Salamanca, a l'Institut de Recherche contre le cancer). Il m'a orienté afin d'aller toujours vers les solutions les plus pratiques, et les compromis les plus acceptables, en terme de performances et temps de développement.

Durant ce stage j'ai travaillé en collaboration avec diverses personnes selon les domaines auxquels se sont appliqués mes travaux. Martha Isabel Goméz Alvarez du département de phytopathologies, a utilisé le logiciel que j'ai développé afin de quantifier la croissance des racines d'une série d'expériences, et d'effectuer l'exploitation statistique des résultats. Rafael Zamudio m'a aidé à définir le travail à effectuer dans la mise à jour de l'application de gestion intégrée de l'oliveraie. Mon maître de stage Fran, ayant développé le système de gestion et de suivi d'échantillons (séquences génomiques) (cf 4.4.1), il m'a particulièrement aidé dans l'élaboration du système d'analyse répartie de séquences.

Chapitre 2

Mesure semi-automatique de la surface de racines

2.1 Besoins, spécifications informelles

2.1.1 Contexte biologique

Trichoderma est un champignon très largement disponible dans la nature, que l'on peut facilement isoler et cultiver, qui se développe très rapidement sur de nombreux substrats[1] sans affecter ni l'homme, ni les animaux, ni les végétaux. Les espèces du genre *Trichoderma* ont été définies comme antagoniste[2] face à d'autres champignons phytopathogènes. Cette propriété fait que de nombreuses souches ont été utilisées comme agents effectifs de *biocontrôle*[3] pour des maladies de cultures ... Par ailleurs certaines ont été brevetées pour leur commercialisation. Les champignons issus du *genre* Trichoderma constituent une alternative biologique plus saine, propre, non accumulable dans la chaine alimentaire, et respectueuse de l'environnement ; en opposition aux pesticides chimiques polluants que l'on utilise habituellement dans l'agriculture. Trichoderma exerce son activité de biocontrôle par différent mécanismes, et fait croître les plantes au niveau radiculaire et végétal.

L'analyse d'images a été utilisée afin de déterminer quantitativement les capacités de croissance des différentes souches de trichoderma sur plusieurs espèces de semences de tournesol. Les semences ont été dans un premier temps "prégermées" puis ensuite inoculées de celles-ci. Durant la période de croissance, des mesures furent réalisées et à la fin de l' expérimentation, les zones radiculaires ont été extraites et disposées sur un plan monochrome afin de prendre les photos.

2.1.2 Outils existants pour la réalisation de cette analyse d'images

Outils commerciaux

Auparavant, les traitements étaient effectués grâce à deux outils, l'un étant dédié à la retouche et au montage photographique, l'autre étant dédié aux SIG[4]. Le seul fait, outre le nombre d'étapes à effectuer et leur paramètrage, de traiter une image par l'emploi de deux logiciels trahi la relative « lourdeur » des opérations.

[1] milieu de culture d'un organisme

[2] Organisme (le plus souvent pathogène) qui n'occasionne pas de dégâts importants mais dont la présence protège ses hôtes des dégâts d'autres organismes nuisibles [NIMP Pub. No. 3, 1996]

[3] réduire les populations d'"ennemis" de culture en faisant appel à leurs ennemis naturels, leurs prédateurs

[4] Systèmes d'Information Géographique

Solution libre de droits : ImageJ

C'est un logiciel issu du domaine public, de traitement et d'analyse d'images scientifiques Il correspond à la réécriture en java du très célèbre NIH Image développé sur Macintosh. Il est disponible à l'adresse *http ://rsb.info.nih.gov/ij/* ainsi que toute la documentation et de nombreux plugins développés par des utilisateurs. Il peut s'éxecuter sur n' importe qu'elle machine disposant de la machine virtuelle Java (ou d'un JRE[5]). Il peut être éxécuté soit en temps qu'*applet*[6] via un navigateur, soit en temps qu'application indépendante. Cette solution de traitement d'images scientifiques s'avère donc être par nature, *multi-platforme*, et conçue pour être *libre* et *ouverte* :

- *Libre* car gratuite
- *Ouverte* pour deux principales raisons : les codes sources sont téléchargeables et documentés, donc par la nature *orientée-objet* de Java facilement modifiables – les traitements sont automatisables à l'aide de *macros* et les fonctionalités d'ImageJ sont aisément extensibles à l'aide de *plugins* (leur élaboration fera l'objet d'un paragraphe par la suite).

Ses fonctionnalités s'étendent du traitement de piles d'images provenant d'équipements d'aquisition d'images médicales, au calcul d'aires et à la détection de particules.

[5] « Java Runtime Environment »

[6] Code java disponible sur le serveur Web, téléchargé sur le navigateur client, et executé au sein d'une page HTML

2.2 Expression du problème, et notions théoriques pour le résoudre

2.2.1 Objectifs de ce système d'analyse d'images

Ce système doit permettre à une personne n'ayant pas de connaissances préalables en imagerie numérique (encore moins en informatique) d'effectuer des mesures à partir de photographies de racines, afin de comparer leur croissance au cours d'expérimentations biologiques. Considérant le logiciel ImageJ, un tel système peut se présenter sous plusieurs formes. La première peut être constituée par un protocole de traitements et d'application de filtres, en utilisant directement les menus de cette application. Ce protocole peut être automatisé à l'aide d'une macro. Cette option ne peut être envisagée que pour des images relativement similaires, en effet, une fois cette macro est lancée, l'utilisateur perd tout contrôle sur la séquence d'opérations réalisées. Afin de lui fournir un contrôle sur les paramètres des traitements, il est possible de développer un *plugin*, qui permettra d'étendre les fonctionnalités d'ImageJ de façon modulaire. Enfin, dans le but de rendre le système spécifique à un type d'images, et plus simple pour l'utilisateur (qui ne souhaite réaliser qu'une seule séquence d'opérations), on pourra substituer l'interface graphique d'ImageJ afin de masquer les contrôles inutiles, et l'adapter ainsi aux spécificités des traitements et au degré d'interaction souhaitée en considérant ImageJ comme une bibliothèque de fonctions de traitement d'images. Ces différentes options résumées sur la figure 2.1 de la page 10.

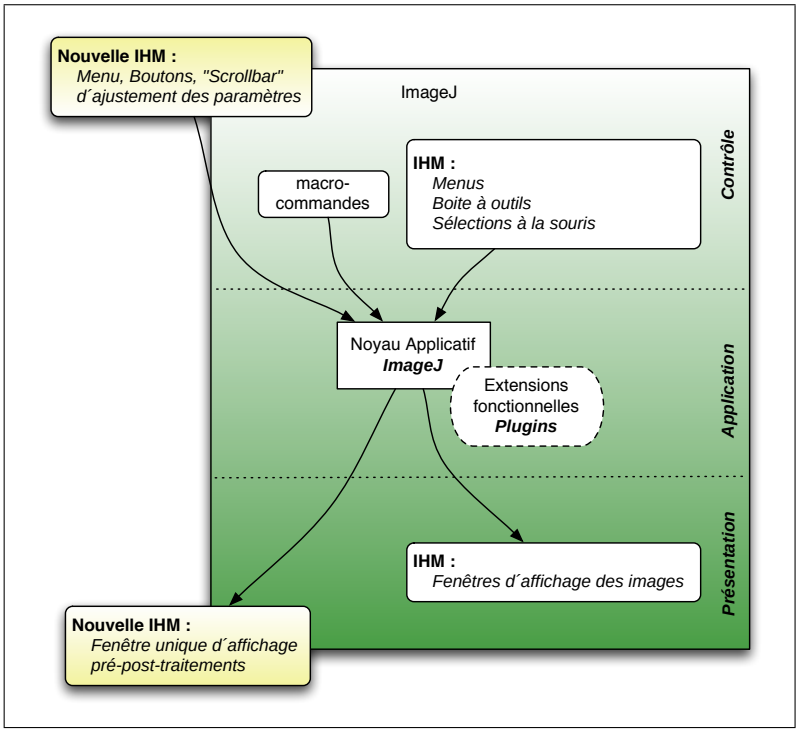

FIG. **2.1** – Du pilotage des fonctionnalités d'imageJ à la présentation des résultats

2.2.2 Notions d'analyse d'image

Dans cette période du stage, où je me suis concentré sur des aspects ayant plus trait au traitement d'image, j'ai pu me documenter sur les notions mises en oeuvre dans de tels systèmes afin de choisir les filtres et algorithmes adéquats. Le problème à résoudre s'inscrit clairement dans le cadre de la *Segmentation* d'images. Son but est d'élaborer une description structurelle de l'image (identifier dans notre contexte chaque entité d'une image, une entité étant caractérisée par une racine). Cette description peut éventuellement prendre la forme d'une image mais également celle de n'importe qu'elle autre donnée caractéristique de ces entités. L'identification consiste finalement à associer à chaque pixel de l'image, une étiquette. Elle doit donc simplifier l'image (c'est à dire sa quantité d'informations) sans réduire de manière trop importante, son contenu. Il existe de nombreuses approches et on sait désormais qu'une segmentation idéale n'existe pas. Elle peut se définir globalement par la partition d'une image I en une ou plusieurs régions $\{R_1, R_2, ..., R_n\}$, ce qui donne formellement :

$$I = \bigcup_{i=1}^{n} R_i, \text{ avec } \forall i, j \in [1, n],\ i \neq j, \quad R_i \cap R_j = \emptyset$$

Une des approches qui a été utilisée dans ce système se base sur le seuillage de l'image.
Les thèmes évoqués par la suite ont fait l'objet de divers tests sur les images qui m'ont été fournies. Finalement, c'est à dire en pratique, les améliorations apportées théoriquement par le filtrage, et la fermeture de l'image binaire n'étaient pas significatives lorsqu'on executait le plugin d'analyse de particules d'imageJ. Par souci de simplicité et d'efficacité, j'ai du les suprimer des traitements à effectuer par l'application.

Pré-traitement par filtrage

Filtrer une image peut se définir comme la convolution de celle ci avec une fonction que l'on appelle *réponse impulsionnelle* ou *opérateur de convolution*. Dans le cas pratique d'une image carrée constituée de pixels (cas discret[7]), l'image filtrée est donnée par : (pour une image de taille n , k étant le domaine de f, et $k \leqslant n$)

$$\begin{aligned} I_f(x, y) &= (f * I)(x, y) \\ &= \sum_{i'=-k/2}^{k/2} \sum_{j'=-k/2}^{k/2} f(i - i', j - j') I(i', j') \end{aligned}$$

Le filtrage linéaire consiste simplement à remplacer chaque pixel (son niveau de gris), par une combinaison linéaire de ses voisins. Le filtre qui a été testé a eu pour but d'accentuer les contours de l'image. C'est un filtre réhausseur, il accentue les hautes fréquences, qui représentent sur l'image les changements rapides d'intensité. L'image est convoluée avec la matrice[8]

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

c'est à dire que la valeur de chaque pixel est multipliée par 5 et on lui soustrait la somme de ses voisins.

[7] pouvant facilement être étendu au cas continu
[8] aussi appelée noyau de convolution

Seuillage

On considère le cadre des images en niveau de gris. Cette étape consiste à segmenter l'image en plusieurs classes en ne se basant que sur son histogramme[9]. A chaque "pic" de l'histogramme, on associe une classe. Plusieurs types d'algorithmes existent pour réaliser cette opération : détéction des vallées de l'histogramme, minimisation de la variance intra-classe, seuillage entropique (par maximisation de l'entropie[10] résultant du découpage suivant plusieurs classes), par maximisation du contraste de l'image...

On se situe dans un contexte où les images sont a priori bi-modales, l'histogramme comporte en effet deux pics, celui caractérisant une entité de l'image (une racine), et celui caractérisant le fond de l'image (il a été choisi de photographier les racines au demeurant "claires" sur un fond sombre afin de faire ressortir clairement ces dernières). L'algorithme qui a été choisi est celui d'Otsu. La méthode peut être résumée ainsi : l'homogénéité d'une région est caractérisée par sa variance (une grande homogénéité implique une faible variance). Le seuil est alors calculé pour qu'il minimise la variance intra-classe.

Détection de contours

Un coutour est une zone de forte variation de niveaux. Un indicateur de cette *pente* peut être la détection d'un extremum de la dérivée, ou d'un passage par zéro de la dérivée seconde. Ces deux approches constituent ce qu'on appelle des méthodes dérivatives(calculs de gradients et laplaciens). De nombreux filtres existent, certains privilégiant une direction pour le calcul du gradient ...

Travail sur les images binaires

Après le seuillage, les structures à identifier étant filaires (racines) des parties de celles-ci, sont parfois déconnectées. Une opération de morphologie mathématique, la "fermeture" permet, à l'aide d'un élement structurant de l'image, de connecter les particules suffisamment proches. Elle consiste à appliquer une *dilatation*, puis une *érosion* avec le même élément structurant. Cette opération permet de connecter des particules proches. L'érosion peut se définir ainsi, on considère X la particule à éroder par l'élement structurant B_x, il s'agit de répondre à la question "B_x est-il contenu entièrement dans X ?" et la traduction formelle serait, Y étant l'érodé de X :

$$Y = \{x \in X \ / \ B_x \subset X\}$$

La dilatation se définit de manière analogue, comme les points de l'image I répondant à la question "B_x touche-t-il l'ensemble X ?" ce qui donne formellement :

$$Y = \{x \in I \ / \ B_x \cap X \neq \emptyset\}$$

[9] c'est à dire la distribution de ses niveaux de gris, en d'autres termes, le nombre de pixels de l'image correspondant à chaque niveau.

[10] mesure de la quantité d'information contenue dans l'image

2.3 Différentes étapes d'élaboration de ce système

2.3.1 Les images

Dans un premier temps il a fallu élaborer les traitements pour des images dificilement exploitables. En effet, les conditions d'illumination n'étaient pas homogènes, et le seuillage via ImageJ ne donnait que des résultats mediocres. Il a donc fallu trouver un moyen pour « bien »prendre les photos. Les biologistes ont pu ainsi extraire du milieu de culture les racines et les disposer sur un plan de couleur sombre. Les photos ont été prises avec un pied et des conditions d'illumination bien meilleures. La figure 2.3.1 de la page 13 montre un exemple de photo prise en début et en fin d'élaboration du logiciel. Ce court paragraphe tend à montrer l'importance de la qualité des échantillons originaux.

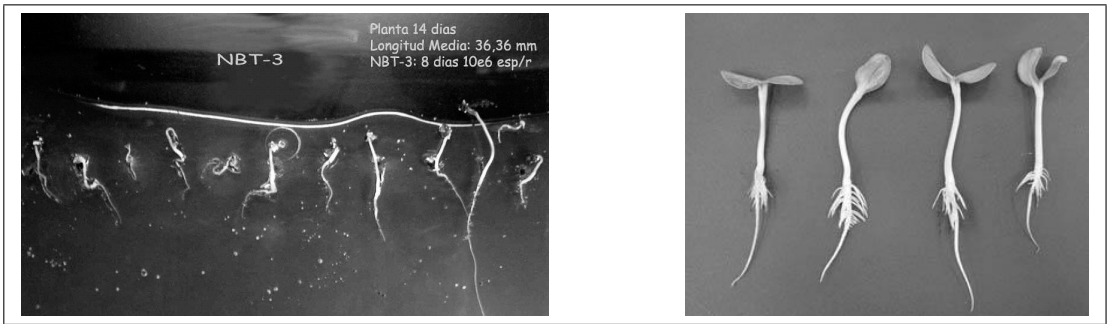

Fig. **2.2** – Images à traiter en début et fin d'élaboration du système de quantification de croissance

2.3.2 Protocole de traitements avec ImageJ

L'élaboration de la suite des opérations à effectuer a fait l'objet de la rédaction d'un court document disponible en annexe à la page 43. En quelques mots, les étapes sont les suivantes :

1. *Conversion de l'image en niveaux de gris*
2. *Accentuation éventuelle des contrastes*
3. *Binarisation de l'image par seuillage*
4. *Application du filtre* `analyse particles`

2.3.3 Développement d'un *plugin* ImageJ

La manière la plus simple pour aborder le développement d'un tel plugin, c'est d'étudier dans un premier temps un exemple simple, il sera par la suite commenté. Seront également détaillés les principaux objets manipulables fournis par l'API d'ImageJ et comment doit être déployé au sein du logiciel, un plugin.

Commençons par un exemple commenté

```
import ij.*;
import ij.gui.NewImage;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
import java.awt.*;
//_____________________________________
public class ZonaInversa implements PlugInFilter{
    //_________________________________
    public int setup(String arg, ImagePlus imp) {
        if (arg.equals("about")) {
            showAbout();
            return DONE;
        }
        // soporta imagenes 8BitsGrays, y Stacks tambien
        return DOES_8G+DOES_STACKS;
    }

    //_____________________________________
    private void showAbout() {
        IJ.showMessage("Enumeracion y Quantificacion de Semillas",
                "blablabla blablabla blablabla");
    }

    //____________________________
    public void run(ImageProcessor ip) {
        int w = ip.getWidth();
        int h = ip.getHeight();
        Rectangle roi = ip.getRoi();
        int offset = 0 ;
        int pos = 0 ; // indice del pixel coriente

        // creacion de la imagen de salida de tratamiento ;
        ImagePlus  inverted = NewImage.createByteImage("Invertida",(int) ip.getWidth(),
                (int) ip.getHeight(), 1, NewImage.FILL_WHITE);
        ImageProcessor inv_ip = inverted.getProcessor();

        inv_ip.copyBits(ip, 0, 0, Blitter.COPY);
        byte [] pixels = (byte[]) inv_ip.getPixels() ;
        // ahorra, estamos trabajando sobre los pixeles de la imagen de salida
        // parsing of the "Region of Interest" y Processing of the pixel
        for (int i = roi.y ; i < (roi.y + roi.height) ; i++) {
            offset = i*w ;
            for (int j = roi.x ; j < (roi.x + roi.width) ; j++){
                pos = offset + j ;
                pixels[pos] = (byte) (255 - pixels[i]) ;
            }
        }
        inverted.show();
        inverted.updateAndDraw();
    } // fin del run
} // fin del plugin
```

FIG. **2.3** – plugin réalisant l'inversion d'une zone prédéterminée de l'image

les packages à importer : Il s'agit du package `ij` pour pouvoir manipuler les objets de base d'ImageJ, `ij.gui.NewImage` pour pouvoir créer de nouvelles images, `ij.process.*` pour les objets de types « *image processor* »qui seront détaillés par la suite, `ij.plugin.filter.PlugInFilter` correspond à l'interface qu'il faut implémenter pour un plugin ImageJ. On importe également le package `java.awt.*` pour manipuler des objets graphiques disponibles avec Java.

la classe du plugin : Elle doit, comme indiqué précédement afin de bénéficier de nombreux services disponibles, implémenter l'interface `PlugInFilter`

la méthode `setup` permet, en fonction des arguments passés en paramètre, de configurer le plugin. Dans notre contexte, on doit implémenter une méthode `showAbout` car on traite le cas où l'argument vaut "about" et pour ne pas effectuer le traitement du plugin (c'est à dire sa méthode `run`), on rend en résultat la constante `DONE`. Dans tous les autres cas, on rend un certain nombre de « *flags*[11] »pour spécifier que le plugin peut traiter correctement des images en niveaux de gris codés sur 8 bits[12], et peut également traiter des piles d'images, de ce même type.

la méthode `run` met en oeuvre les traitements effectués par le plugin. On récupère dans un premier temps le processeur associé à l'image sur laquelle on applique ce plugin. On peut alors disposer des informations sur les dimensions de l'image et sur la sélection effectuée par l'utilisateur[13] : `ip.getWidth()`, `ip.getHeight()`, `ip.getRoi()`. Il faut par la suite déclarer et initialiser les objets que l'on souhaite en sortie de filtre : dans notre cas une nouvelle image correspondant à l'objet `ImagePlus` nommé ici *inverted*, on récupère également son processeur associé pour effectuer les traitements sur cette nouvelle image. On copie alors les pixels de l'image originale dans la nouvelle avant d'effectuer l'inversion (méthode `copyBits()` de la classe `ImageProcessor` dont la signature est détaillée dans la documentation de l'API d'ImageJ). Cette première phase étant réalisée on dispose alors d'une copie des données qui nous sont disponibles en entrée du filtre et sur lesquelles on peut désormais travailler, et celà, sans risque de corruption des données originales. Le traitement effectué est le suivant : on dispose du tableau de pixels (l'adressage des pixels ne sera pas détaillé[14]), à partir du processeur associé à l'image de sortie (`byte [] pixels = (byte[]) inv_ip.getPixels()`). Il suffit ensuite de parcourir ce tableau (chaque ligne de la region d'intérêt) avec deux boucles imbriquées et d'inverser chaque pixel. On conclue en affichant l'image par les deux méthodes suivantes : `show()` et `updateAndDraw()` .

Principaux concepts d'ImageJ

Images elles sont représentées par les objets `ImagePlus`, leur type[15] est spécifié par une constante, on peut lui donner un titre, accèder à sa taille ...

Processor c'est l'objet sur lequel on peut manipuler l'image à laquelle il est associé, lorsqu'on souhaite appliquer des filtres, ou directement travailler sur les pixels de l'image. Il est nommé `ImageProcessor`.

Région d'intéret c'est un rectangle de type `java.awt.Rectangle` qui permet de limiter la zone d'action pour les filtres et les différentes opérations qu'on souhaite réaliser.

[11] configuration binaire caractéristique d'un comportement que l'on veut donner à l'application
[12] 8 bits correspondent à 256 niveaux de gris possibles
[13] on parle alors de ROI pour « *Region Of Interest* »
[14] à savoir simplement que l'on est en présence d'un tableau de pixels unidimensionel
[15] image couleur sur 8bits, gris sur 8, 16 ou 32 bits ...etc ...

Déploiement du plugin

Les plugins développés par les utilisateurs doivent être situés dans un dossier nommé `plugins`, qui est un sous-dossier d'ImageJ. Les sous dossiers situés dans ce dossier `plugins` sont alors affichés dans le menu du même nom. Pour installer un plugin (une classe Java compilée, et ainsi suffixée par `.class`), il suffit de copier le fichier `.class` dans le dossier `plugins`[16]. Eventuellement, si l'utilisateur se trouve en possession du code source du fichier, il devra le compiler au préalable à l'aide par exemple du compilateur fourni par le JDK[17] en éxécutant par exemple dans un terminal `javac MonPlugin.java`.

Le plugin dévelopé pour l'analyse des racines

Il a été réalisé suivant le modèle de l'exemple précédent. On effectue simplement dans le corps de la méthode `run()` les différents appels aux fonctionnalités d'ImageJ afin de réaliser le protocole de la page 13. à détailler .... Toutefois, ce plugin est resté à l'état de test car le besoin d'interaction avec l'utilisateur s'est avéré nécessaire afin de superviser les mesures. De plus les traitements effectués par ce dernier ont évolué vers cette forme (résumé par le schéma de la figure 2.4), pilotés par une interface graphique décrite en section 2.3.4.

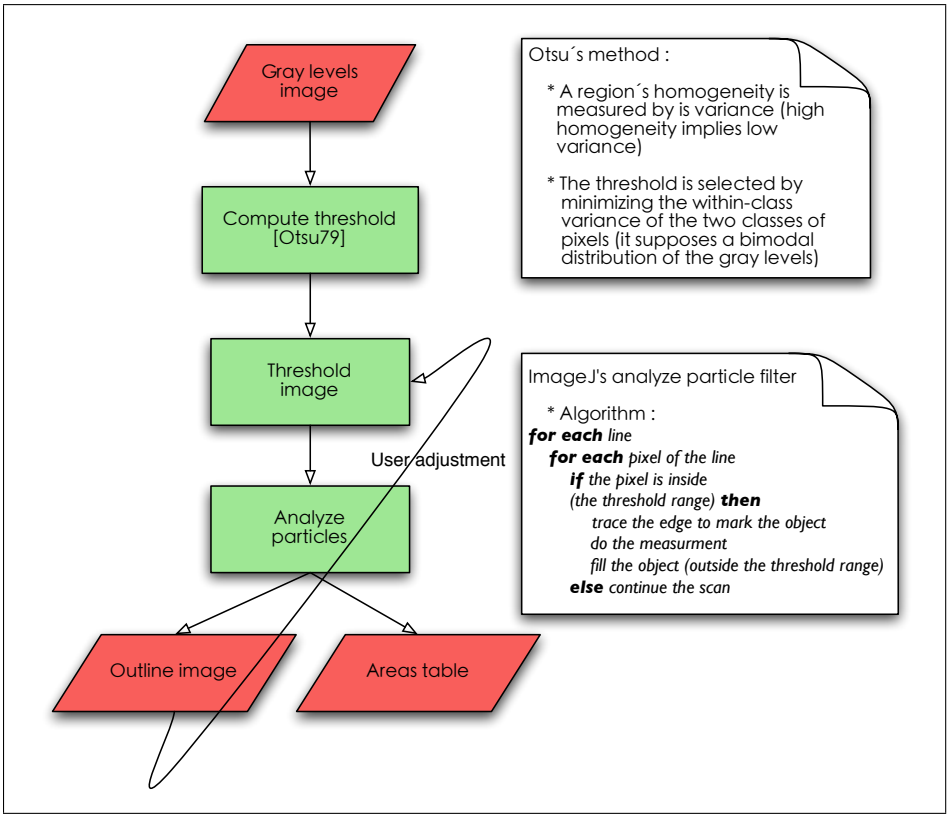
Gray levels image
Compute threshold [Otsu79]
Threshold image
User adjustment
Analyze particles
Outline image
Areas table
Otsu´s method :
* A region´s homogeneity is measured by is variance (high homogeneity implies low variance)
* The threshold is selected by minimizing the within-class variance of the two classes of pixels (it supposes a bimodal distribution of the gray levels)
ImageJ's analyze particle filter
* Algorithm :
for each line
for each pixel of the line
if the pixel is inside
(the threshold range) then
trace the edge to mark the object
do the measurment
fill the object (outside the threshold range)
else continue the scan

FIG. **2.4** – suite des traitements appliqués à l'image à traiter

2.3.4 Développement d'une IHM

Le temps imparti pour la réalisation de ce système d'analyse d'images a été relativement court. Ayant auparavant réalisé un stage avec un sujet similaire au sein de l'université[18], j'ai

[16] ou dans un de ses sous-dossiers
[17] « Java Development Kit »
[18] sous la direction de Patrick Bron

pu m'inspirer fortement de l'interface graphique que j'avais réalisé à cette époque.

Design

Une copie d'écran de l'interface (figure 2.5 de la page 17) permet d'entrevoir les principales fonctionnalités de l'application et les possibilités d'interaction offertes à l'utilisateur.

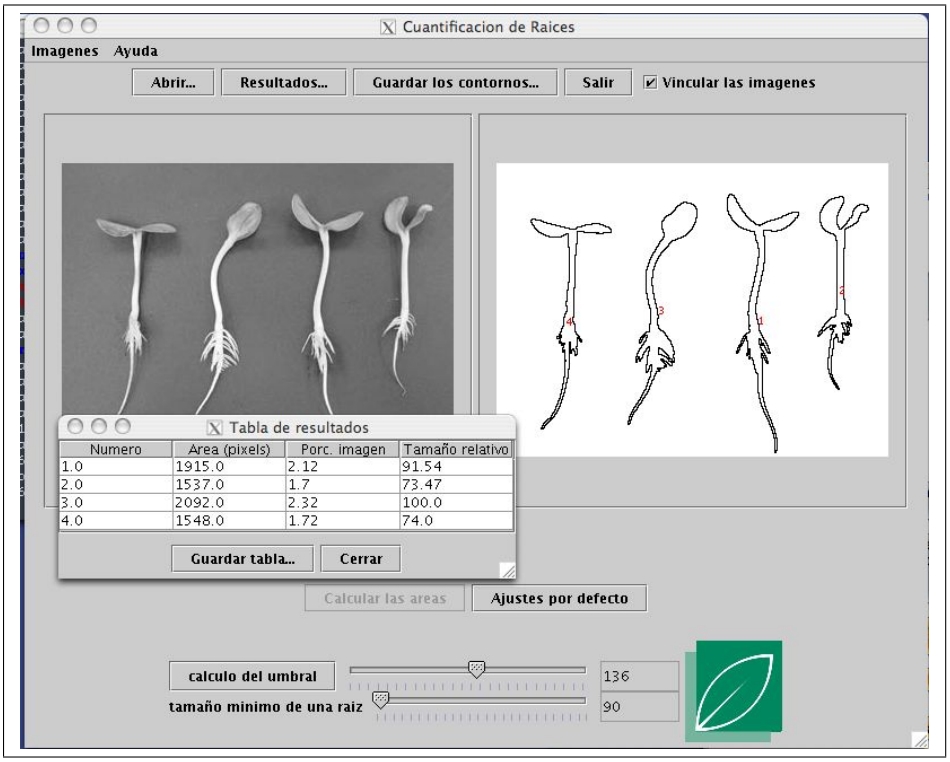

FIG. **2.5** – IHM de l'application de quantification de croissance de racines

En voici une succinte description :

affichage : La fenêtre principale est divisée en trois zones principales : la zone supérieure comporte des boutons pour contrôler l'application ; la zone médiane se charge d'afficher dans deux composants *JPanel*, l'image avant et après traitement (leur défilement peut être lié afin de voir la correspondance entre les deux images si on veut controler la détection des contours des racines) ; la zone inférieure est constituée par deux *JScrollBars* qui permettent de contrôler les paramètres du traitement.

contrôle du traitement : il s'effectue à l'aide de boutons qui sont, selon le contexte, inactifs, d'un menu, et de barres de réglage.

résultats d'analyse : ils sont affichés dans une fenêtre séparée via un composant *SWING JTable*. Il est par ailleurs possible d'exporter cette table dans un fichier *ASCII* que l'on peut importer directement dans un quelconque tableur.

assistance à l'utilisateur : elle est carcatérisée par des *JTooltips* sur les objets de contrôle le nécessitant et par une fenêtre d'aide comportant un petit manuel d'utilisation.

L' annexe A.1 de la page 42 illustre, avec un diagrame de séquence UML, la durée de vie et les comunications inter-objets lors du traitement d'une image.

2.4 Mise en perspective, regard critique ...

Retours quant à l'utilisation du nouveau système

Ils ont été relativement positifs, et ses avantages ont étés formulés ainsi :

- *rapidité :* il faut maintenant moins de 5 minutes pour traiter chaque image, il en fallait plus d'une vingtaine avec l'ancien protocole, utilisant deux logiciels.
- *simplicité :* tous les traitements sont intégrés et automatisés au sein d'une interface graphique constituée d'une seule fenêtre. Un seul paramètre est ajustable, le seuil de binarisation.
- *ouverture :* les résultats de la segmentation sont exportables sous la forme d'une table ascii, facilement importable dans un tableur pour traitements statistiques ultérieurs.

Il a par ailleurs été suggéré la réalisation de zoom, afin de manipuler des images de grande taille.

Défauts de l'algorithme

L'algorithme de détection de contour d'ImageJ a été mis en défaut lorsqu'il y a des recouvrements sur l'image, en d'autres termes, lorsque les racines se croisent. En voici une illustration :

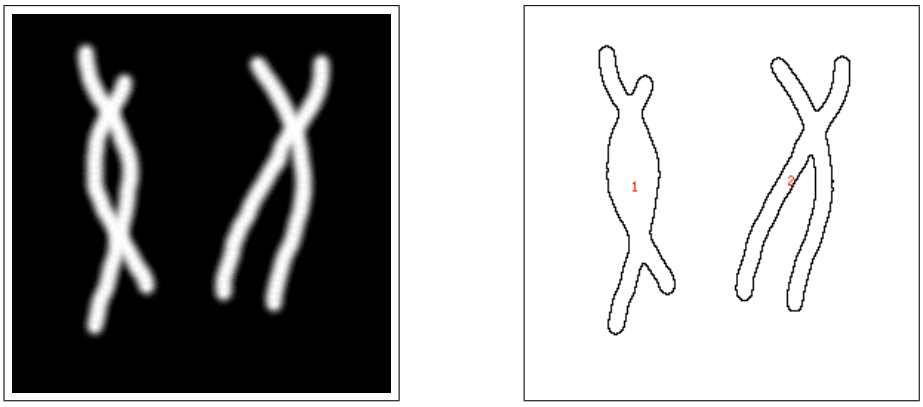

FIG. **2.6** – dessins illustrant les défauts de l'algorithme

Une solution, consisterait à énumérer les pixels se trouvant sous, ou sur le seuil déterminé, on passerait donc outre ce problème de zones englobées. Cette idée n'est malheureusement pas satisfaisante car on n'effectuerait qu'une mesure globale sur l'image, perdant la possibilité de comparer plusieurs entités sur la même image. Supposant une image bi-modale constituée dune classe de pixels *Fond* et d'une seconde *Entités*, une solution alternative consisterait à développer un algorithme hybride, mélangeant l'approche détection de particule d'imageJ, avec une soustraction des pixels englobés dans une entité, mais appartenant à la classe *Fond*. Le problème à résoudre est de pouvoir identifier un pixel appartenant à une région, sans avoir de paramétrisation mathématique du contour de celle-ci. Une réponse consisterait en une décomposition en composantes connexes de l'image binaire, c'est à dire en associant à une composante les pixels d'une même classe "connectés" entre eux. Il a été choisi de se contenter des erreurs induites par l'algorithme de détection d'imageJ.

Fonctionalités à ajouter

Deux fonctionnalités déjà présentes au sein d'imageJ seraient intéressantes à ajouter à l'application :

1. Un *zoom* : qui permettrait de travailler sur des images avec une résolution élevée, bien que pour l'instant, effectuer des mesures extrêmement précises n'est pas à l'ordre du jour, on se contente en effet de faire des comparaisons entre racines.
2. Un système de *calibration* : en ajoutant un réglet (c'est à dire un étalon) à la photo, le plugin d'imageJ réalisant l'analyse de particules peut le prendre compte après avoir indiqué à l'aide d'un segment que dessine l'utilisateur sur l'écran, la mesure d'1cm (par exemple). Les résultats de calculs d'aire ne seront alors plus exprimés en pixels, mais en cm^2.

Chapitre 3

Maintenance et mise à jours d'une application web

3.1 Contexte

La culture des oliviers est un des secteurs prépondérent de l'agriculture andalouse ; en effet, les conditions météorologiques y sont particulièrement propices et la composition du sol favorable. C'est pourquoi une demande technologique d'aide à la décision, en termes d'irrigation et d'additifs nutritifs peut croître de manière importante. C'est la raison d'être du service/produit GIO de NBT.

Cette application constitue donc le coeur du service de gestion intégrée de l'Oliveraie. Elle est caractérisée par ce qu'on appelle une "*application web*" se situant dans le cadre des applications informatiques trois tiers. En d'autres termes, elle se décompose en trois couches communicantes par échange de services :

1. *un serveur de bases de données* : les données sont stockées dans une base PostGreSQL, et accédées via des requêtes SQL[1].
2. *un serveur d'application "middleware"* : c'est dans notre contexte un conteneur d'applications web Tomcat. Il constitue en quelques sortes un serveur Web amélioré car en plus de diffuser des pages Web, il peut aussi exécuter du code Java (afin d'interagir avec l'utilisateur) au sein de pages JSP[2], et de créer ces pages dynamiquement.
3. *le client* : peut être vu sous différents angles, sous un côté plus humain, on peut considérer que c'est la personne se situant derrière son navigateur web, consultant les pages de l'application. C'est l'entité "*demandeur*" de ressources.

L'application se décompose au niveau "interface homme-machine" en deux parties. Une permettant à l'administrateur de la faire fonctionner, l'autre, permettant à l'utilisateur de souscrire au service en s'enregistrant, et en caractérisant ses parcelles le plus précisément possible. Techniquement, ces pages sont réalisées à l'aide des technologies JSP, Servlet[3], HTML et JavaScript.

Le fonctionnement de l'application peut être résumé de la sorte :

1. Le client s'enregistre, et caractérise ses oliveraies.

[1] Structured Query Langage
[2] Java Server Pages
[3] code java s'exécutant côté serveur

2. Chaque mois, l'administrateur met à jour la base de données météorologiques (températures, précipitations, nombre de jours en dessous et au dessus d'une certaine température...) à l'aide d'un programme externe à l'application.
3. Quand il le souhaite, le client peut solliciter auprès de l'administrateur, le calcul d'une prédiction sur le mois à venir, basée sur les données météorologiques de l'Andalousie depuis une cinquantaine d'années, et constituant ainsi une aide aux décisions à prendre en terme d'irrigation et de nutrition des arbres.

3.2 Problèmatique

Cette application a été développée par une autre entreprise, et j'ai pu constater qu'il s'agit manifestement d'une version en cours de développement, ayant une taille avoisinant les 30000 lignes de code, non commenté. Par ailleurs, les documents de spécification et de conception ne sont pas en possession d'NBT, et les contacts avec cette entreprise sont inexistants. Le problème est donc le suivant : prendre le temps de réaliser une "cartographie" de cette application, c'est à dire représenter les flux de données entre les pages JSP/HTML. C'est un travail qui m'a semblé nécessaire si on souhaite pouvoir localiser, et traçer les *bugs* éventuels.

C'est aussi une étape qui m'a semblé nécessaire afin de localiser où doivent s'appliquer les mises à jour. La plus importante de celles-ci consistera à intégrer à la partie d'administration, les fonctionnalités d'importation de données météorologiques dans la base de données.

3.3 Travail réalisé

3.3.1 Documentation et maintenance

La réalisation de cette documentation à nécessité la recherche de tous les liens possibles entre les différentes composantes de cette application, qui sont des pages appartenant aux deux parties *client* et *administration*, et des *beans* Java. Le formalisme de représentation qui a été induit par cette recherche, est celui des graphes. Un premier a été réalisé mais sa complexité le rend dificilement exploitable (constitué de 128 états, et de 368 transitions), il a fallu le scinder en deux, un axé sur les aspects plus applicatifs, l'autre sur des aspects communication. La figure 3.1 de la page 21 est la légende de ces graphes :

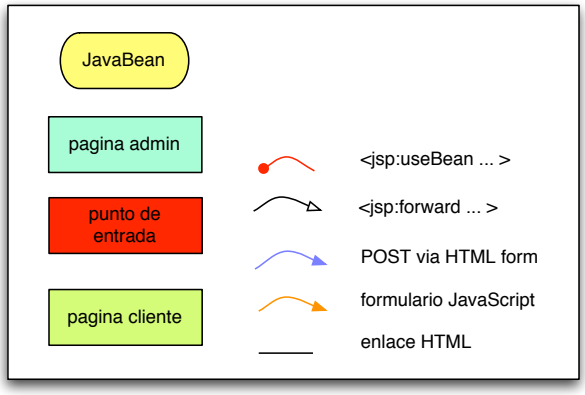

FIG. **3.1** – légende des graphes représentant l'application web

Sont donc représentés dans ces graphes, ses points d'entrée, les pages "clientes" et "administration" de l'application, et l'application proprement dite, c'est à dire les *JavaBeans* éxécutés sur le serveur.

Les figures (B.2, B.1 en pages 50 et 49) disponibles en annexe sont des extraits des deux graphes issus de la scission du premier. La réalisation de ces graphes a été une étape relativement longue car il a fallu examiner le code et les réaliser à la main. J'imagine que l'écriture de scripts permettant la construction de graphes à partir des pages jsp doit être possible (avec des outils comme GraphViz). Par ailleurs des outils commerciaux réalisant ce travail doivent certainement exister, mais le temps consacré à cette recherche s'est avéré trop mince.

La maintenance de cette application a consisté principalement à du "nettoyage" de code, c'est à dire à la supression des doublons de fichiers figurant, et dans la partie *client*, et dans la partie *administration*, et de certaines parties de code en commentaire. Les autres tâches de maintenance ont consisté à effectuer des modifications d'interface graphique, par l'ajout de *ToolTips*[4], et à imprimer dans la page affichée par le navigateur les exceptions Java en cas de bug du système. En effet auparavant, seul un message non informatif de type *"Erreur Système, contacter l'administrateur"* était affichée.

3.3.2 Ajout de fonctionnalités

Cette nouvelle version de l'application GIO, est principalement caractérisée par l'ajout au niveau de la partie administration de l'interface utilisateur, d'une page JSP permettant l'envoi vers le serveur hébergeant celle-ci, des fichiers[5] de relevés météorlogiques fournis par l'institut national de météorologie[6]. Un *bean*[7] Java, réalise le *parsing* du fichier et l'envoi des données via des requêtes SQL. Ce *bean* Java a été développé à l'aide de l'application Java externe réalisant ces mêmes fonctions. Il a fallu dans un premier temps, comprendre le fonctionnement de celle-ci afin de l'intégrer à l'application web sous la forme d'un fichier *jar* fournissant des bibliothèques de fonctions. Elles comportent la possiblité de *parser* des fichiers au format Microsoft Excel, de connection à une base de données PostGre via pont JDBC[8]. Ce *bean* a donc été intégré a l'application, et est instancié lors de l'ouverture de la page JSP.

L'envoi des fichiers a été réalisé par l'achat d'un autre composant java (lui aussi sour la forme *JavaBean*) permettant de gèrer l'upload de fichiers vers un serveur , et a consisté á son intégration au sein de l'application via la même page JSP.

3.4 Mise en perspectives

Plusieurs choses restent à explorer au sujet de l'évolution de cette application, comme la conformation aux spécifications de *Sun* : les *Enterprise Java Bean (EJB)*. Je me suis documenté à ce sujet, et ils permettent de développer plus aisément les parties ayant trait aux sessions utilisateurs, à la gestion des bases de données associées, à la sécurité... Mais ceci sous entend un travail relativement lourd de refonte complète de l'application.

Un autre thème intéressant et plus facilement réalisable serait de travailler sur un format d'échange de données météorologiques, basé sur les technologies XML, afin de faciliter l'échange de données structurées, en définissant par exemple un shéma XSL, modélisant la structure de telles données. Le problème se posera effectivement le jours où l'*INM* changera sa façon de présenter les données météorologiques (à l'heure actuelle sous forme de fichiers au format Microsoft Excel 97). Il faudrait par ailleurs entrer en contact avec l'*INM* afin

[4] indication textuelle qui s'affiche en fonction de la zone survolée par la souris
[5] au format Microsoft Excel
[6] INM, instituto nacional de meteorología
[7] objet Java respectant les spécifications de *Sun* ; sous cette forme, il devient un composant réutilisable
[8] *Java Database Connectivity*

de savoir s'ils ne travaillent pas déjà avec les technologies XML.

Les résultats de projections sont affichés dans des tables HTML, qui ne sont pas destinées à être imprimées sur du papier au format *A4*. En effet, ce que présente l'administrateur au client, est un rapport sous forme "papier". J'ai réfléchi à une solution simple consistant à profiter de la construction dynamique des pages(par génération de balises HTML au sein des pages JSP) pour construire un fichier à balises au format *LaTex* sur le serveur. Il peut être par la suite traduit au format de documents portables PDF (ou PostScript) qui sont eux directement destinés à l'impression sur papier. Les tests effectués se sont avérés satisfaisants, mais n'ont pas étés menés à terme, mon travail sur cette application Web étant considéré comme terminé. Cette solution semble prometteuse bien que le coût de la traduction en PDF / PS sur le serveur soit à prendre en compte.

Chapitre 4

Comparaison répartie de séquences génomiques

4.1 Introduction thématique

Le travail réalisé s'insère dans le cadre du projet EstASA : « *Semi-automated sequence analysis pipeline for ESTs' projects* » dont en voici un court résumé. Les maladies affectant les plantes causent chaque année de très sévères pertes dans les récoltes . Les cultures peuvent être efficacement protégées au moyen d'agents de contrôle biologique, comme par exemple *Trichoderma*[1], réduisant ainsi de manière drastique l'usage des pesticides. EstASA est un système d'annotation et d'analyse automatique de séquences, centré principalement sur les projets de séquençage d'ESTs[2]. NBT coordonnant le projet internationnal de génomique fonctionnelle TrichoEST, son laboratoire de biologie moléculaire effectue de nombreuses tâches de séquençage. Une des principales activité du département de bioinformatique est de mettre en oeuvre des structures et des outils ayant pour but de stocker et d'analyser la masse de données générées par ce type de projets.

4.1.1 Quelques notes de Biologie

Afin de pouvoir comprendre la terminologie, cette section comportera quelques courts paragraphes traitant des notions biologiques manipulées au cours de ce stage et s'adresse donc a des gens qui, comme moi, ne sont pas spécialistes en génétique.

Par analogie à un programme informatique, on peut considérer que l'information génétique est le "programme" régissant le fonctionnement des cellules. C'est en effet le code permettant à un organisme de synthétiser des protéines. Le « Dogme Central de la Biologie Moléculaire (3) » a été introduit à la fin des années 1950 par Francis Crick[3], et établit le lien entre l'information génétique détenue par une cellule et ses fonctionnalités, c'est à dire, les protéines qu'elle peut synthétiser. En voici un très simple résumé :

Dans un premier temps, la *réplication* conforme de la molécule d'ADN lors de la multiplication cellulaire permet de conserver le matériel génétique (donc l'identité biologique). On dit que c'est un support stable et transmissible de l'information génétique. Par la suite, afin de synthétiser une protéine, un fragment de cet ADN (que l'on appelle communément un gène) est *transcrit* en *ARN* messager puis va s'associer au ribosome(ARNr) permettant

[1] qui a été présenté en section 2.1.1 de la page 8
[2] pour *Expressed Sequence Tag*
[3] co-découvreur de l'ADN

la synthèse des protéines. On voit donc clairement trois étapes shématisées en page 25.

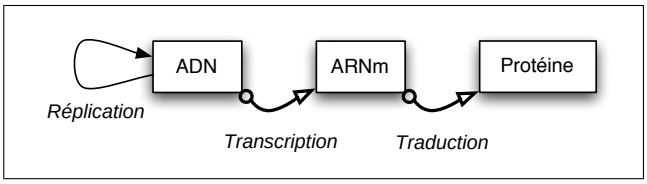

FIG. **4.1** – dogme central de la biologie moléculaire

ADN : acide désoxyribonucléique, molécule en forme de double-hélice. C'est le "matériel" héréditaire qui se transmet pour un organisme de générations en générations. Cette molécule constitue le langage régissant la synthèse des protéines. Il est complexe mais son alphabet est simple[4]. Il comporte quatre unités lexicales (si on reste avec une vision "*langage*") : adénine(A), cytosine(C), guanine(G), thymine(T). Si on déroule virtuellement cette double hélice, la molécule se présente alors comme 2 séquences de nucléotides superposées, ces dernières étant liées deux a deux : A/T et G/C. On peut donc déduire d'une branche sa duale. C'est un mécanisme qui permet la réplication conforme. Il n'y a pas de perte d'information si on ne considère qu'une seule branche, c'est pour cela qu'en pratique on ne travaille que sur une séquence.

Gène : C' est l'unité fonctionnelle de la séquence, en d'autres termes, les instructions (et régulations) pour la production de protéines. Il peut être vu comme un mot du langage.

ARN : molécule proche de l'ADN, mais constituée d'une seule séquence de nucléotides, ayant à la place de la thymine, l'uracile(U). Il en existe de plusieurs types comme l'*ARNm*, jouant le rôle de messager dans le processus de synthèse de protéines.

protéines : constituent les édifices et "machineries" au sein d'une cellule. Elles sont chimiquement très différentes d'une molécule d'ADN ou d'ARN : ce sont des séquences d'acides aminés (structure unidimensionnelle vu sous cette forme) déterminant la configuration spatiale de celle ci (structure tridimensionnelle), et ainsi ses fonctionnalités.

génome : est constitué par l'ensemble de l'ADN codant et non codant d'un organisme,

Avec ces quelques notes, on peut désormais expliquer ce qu'est un **EST** et comment il est obtenu. On sait que grossièrement, 95% du matériel génétique n'est pas réellement codant. Donc les gènes se situent dans les 5% restant. Pourquoi alors séquencer l'intégralité de l'ADN d'un organisme (processus par ailleurs relativement couteux), si on ne s'intéresse qu'à son génome exprimé ? Un gène étant caractérisé par son *ARNm* il s'agit alors (cf figure 4.2) de l'isoler et de réaliser sa transcription inverse, on obtient alors de l'*ADNc*. Le clonage et séquençage de ce dernier permettent de produire une séquence exprimée, que l'on nomme communément *EST*.

4.1.2 BLAST : Alignements et Similarités de séquences

C'est un acronyme pour *Basic Local Alignment Search Tool*, qui constitue une « suite logicielle » qui est devenue une pièce maîtresse dans les domaines couverts par la Bioinformatique. Son principal intérêt est de rechercher la similarité entre des séquences biologiques[5] afin d'identifier celles qui sont inconnues. Plusieurs raisons l'ont amené à un tel niveau de popularité :

– sa *rapidité* : les projets de séquençage de génomes entiers sont devenu nombreux. Il est maintenant nécessaire de réaliser ces calculs de manière massive.

[4] il existe également des lettres décrivants les ambiguités

[5] On entend par « séquence biologique » tant les séquences nucléiques (ADN, ARN) que protéiques

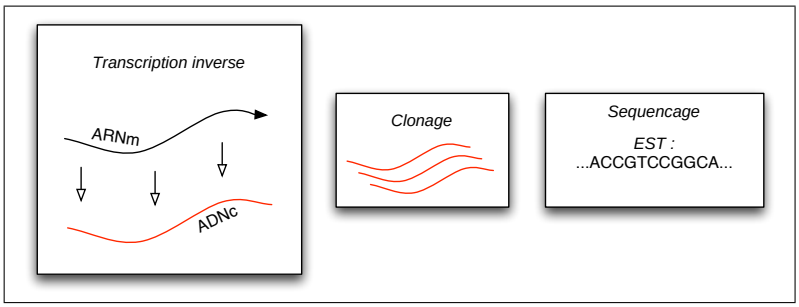

FIG. **4.2** – processus de séquencage d'un *EST*

- sa *fiabilité* : elle peut être caractérisée tant sur un plan statistique que sur des points de vue pûrement informatiques()
- sa *flexibilité* : il existe en effet une famille d'applications travaillant sur differents types d'entrées et de bases de données

Blast évalue de manière statistique la similarité entre les séquences. Pour cela, Blast réalise un alignement entre les séquences deux à deux ; certains programmes sont quant à eux spécialisés dans la réalisation d'alignements multiples.

Alignement de séquences

On se trouve en présence de deux types de données : les deux séquences à aligner, et une matrice de scores, permettant d'évaluer cet alignement. L'évaluation fera l'objet du paragraphe suivant. Le but d'un alignement (global[6]) de deux séquences est, outre son évaluation, , de mettre en évidence la meilleure correspondance entre des résidus de celles-ci, en conservant bien sur l'ordre. Les mutations que l'on peut observer sur les séquences sont les *substitutions, insertions et délétions*. On peut par exemple chercher à aligner les deux séquences suivantes : `PROBLEME` et `POEME` ce qui pourrait donner par exemple :

séquence 1	P	R	O	B	L	E	M	E
séquence 2	P	-	O	-	-	E	M	E

Les algorithmes permettant de calculer ces alignements se fondent sur la "*Programmation Dynamique*" qui est souvent utilisé dans le cadre des problèmes d'optimisation. Ce principe (sous certaines contraintes[7]) consiste à explorer l'espace des sous-problèmes, et mémoriser les solutions optimales de ces derniers. On trouve la solution du problème global en effectuant un "*backtracking*". En pratique les choses se passent ainsi : On crée une matrice 2D permettant d'apparier les résidus de séquences : que l'on remplit à l'aide d'une matrice de score et d'une fonction de pénalité évaluant les insertions ou délétions ("*gap*"). Cette matrice est de plus enrichie par des pointeurs permettant d'effectuer le "*backtracking* : insertion $\leftarrow$, délétion $\uparrow$, Match/Mismatch $\nwarrow$.

1. Données : matrice de scores s, fonction de pénalités g, et les deux séquences
2. Initialisations :
   $M_{(0,0)} = 0$;
   $M_{(0,j)} = g * j$;
   $M_{(i,0)} = g * i$;

[6] il existe en effet deux types d'alignements, locaux et globaux

[7] comme par exemple l'expression de la solution optimale du problème en fonction des solutions optimales des sous-problèmes

3. Remplissage :
   on considère la matrice de score simplifiée suivante :

$$s_{(i,j)} = \begin{cases} 1 & \textit{si les unités syntaxiques correspondent : "Match"} \\ 0 & \textit{s'il y a substitution : "Mismatch"} \end{cases}$$

   et la fonction de pénalité constante $g = -1$.

$$M_{(i,j)} = max \begin{cases} M_{(i-1,j-1)} + s_{(i,j)} & \textit{enrichie du pointeur} \nwarrow \\ M_{(i-1,j)} + g & \textit{avec le pointeur} \quad \uparrow \\ M_{(i,j-1)} + g & \textit{avec le pointeur} \quad \leftarrow \end{cases}$$

4. *"Backtracking"* : permet de déterminer un alignement parmi tous les optimaux en fonction de la matrice M. A partir de la dernière cellule remplie (dans le cas d'un alignement global), on remonte vers la voisine en s'aidant des pointeurs mémorisés. On remarquera par la suite que plusieurs alignements sont à considérer si plusieurs pointeurs sont mémorisés.

Voici un d'alignement global pour les deux séquences précédentes `PROBLEME` et `POEME`, avec l'algorithme de Needlman & Wunsch :

		P	R	O	B	L	E	M	E
	0	−1	−2	−3	−4	−5	−6	−7	−8
P	−1	$\nwarrow$ 1	$\leftarrow$ 0	$\leftarrow$ −1	$\leftarrow$ −2	$\leftarrow$ −3	$\leftarrow$ −4	$\leftarrow$ −5	$\leftarrow$ −6
O	−2	$\uparrow$ 0	$\nwarrow$ 1	$\nwarrow$ 1	$\leftarrow$ 0	$\leftarrow$ −1	$\leftarrow$ −2	$\leftarrow$ −3	$\leftarrow$ −4
E	−3	$\uparrow$ −1	$\uparrow\nwarrow$ 0	$\nwarrow$ 1	$\nwarrow$ 1	$\nwarrow\leftarrow$ 0	$\nwarrow$ 0	$\leftarrow$ −1	$\nwarrow\leftarrow$ −2
M	−4	$\uparrow$ −2	$\uparrow\nwarrow$ −1	$\uparrow\nwarrow$ 0	$\nwarrow$ 1	$\nwarrow$ 1	$\nwarrow\leftarrow$ 0	$\nwarrow$ 1	$\leftarrow$ 0
E	−5	$\uparrow$ −3	$\uparrow\nwarrow$ −2	$\uparrow\nwarrow$ −1	$\uparrow\nwarrow$ 0	$\nwarrow$ 1	$\nwarrow$ 2	$\leftarrow$ 1	$\nwarrow$ 2

Tab. 4.1 – Remplissage de la matrice de programmation dynamique avec l'algorithme de Needlman & Wunsch

Le tableau suivant illustre le *Backtracking*, on retrouve bien l'alignement effectué à la main (en page 26). Une flèche horizontale représente une insertion sur la séquence de l'axe des ordonnées[8].

		P	R	O	B	L	E	M	E
	0								
P		$\nwarrow$ 1	$\leftarrow$ 0						
O				$\nwarrow$ 1	$\leftarrow$ 0	$\leftarrow$ −1			
E							$\nwarrow$ 0		
M								$\nwarrow$ 1	
E									$\nwarrow$ 2

Tab. 4.2 – Backtracking pour determiner l'alignement global

Les deux tableaux suivant illustrent la détection du meilleur alignement local ave l'algorithme de Smith & Waterman. Il est réalisé suivant le même principe. Seules quelques modifications interviennent lors de la phase d'initialisation : la première ligne et première

[8] réciproquement, une flèche verticale représenterait une insertion dans la séquence de l'axe des abscisses

colonne de la matrice sont initialisées à 0, les scores sont supérieurs ou égaux à 0, et le pointeur est mémorisé si et seulement si on a $[s > 0]$.

		P	R	O	B	L	E	M	E
	0	0	0	0	0	0	0	0	0
P	0	↖ 1	0	0	0	0	0	0	0
O	0	0	0	↖ 1	0	0	0	0	0
E	0	0	0	0	0	0	↖ 1	0	↖ 1
M	0	0	0	0	0	0	0	↖ 2	← 1
E	0	0	0	0	0	0	↖ 1	↑ 1	↖ 3

TAB. 4.3 – matrice de l'algorithme de SMITH & WATERMAN

Le meilleur alignement local entre les séquences est obtenu en effectuant le backtracking en ne commençant pas forcément par la dernière cellule remplie mais par celle ayant le meilleur score. Ce qui nous donne ici la sous-séquence commune `EME`.

On peut remarquer que ces deux algorithmes ont l'avantage d'être exacts, mais sont extrêmement coûteux en mémoire et temps de calcul : ces coûts sont en effet proportionnels au produit de la taille des séquences, on a une complexité en $O_{(m*n)}$ si m est la taille de la première séquence et n la taille de la seconde. A noter qu'il existe des implémentations « linéaires en mémoire » mais on est loin de la rapidité de **Blast**. Elle est en effet due à une approche heuristique (4) consistant a trouver rapidement un court alignement exact (en fait un mot qui sera recherché dans les bases de données). Les séquences répondant à cette requête forment les "*hits*". Dès lors, Blast construit systematiquement l'alignement complet par programmation dynamique et effectue un seuillage sur le score de ceux-ci. Il rend donc des séquences suffisamment similaires, mais ne garanti pas de trouver la séquence ayant l'alignement optimal. C'est le prix a payer lorsqu'on cherche la "rapidité"[9].

Similarité entre séquences

Elle est évaluée à l'aide d'une matrice de scores, et de divers indicateurs staistiques. On a vu que Blast était en fait une famille de logiciels, certains travaillent sur des séquences d'ADN, d'autres sur des séquences de protéines, les comparaisons ayant lieu sur des banques d'ADN ou de protéines. Celui que nous utilisons traduit la séquence d'ADN en toutes les séquences d'acides aminés possibles et les compare à une banque de protéines. La matrice de score utilisée s'appelle `BLOSUM62` et attribue un score de similarité pour chaque paire possible d'acide aminé. C'est elle qui est utilisée dans les algorithmes de comparaison par programmation dynamique($s_{i,j}$). Quelques indicateurs de similarité :

Score de l'alignement (S) : somme des valeurs données par la matrice de substitution (`BLOSUM62`) en soustrayant les pénalités d'insertion (*gap*).

Bit score : score normalisé permettant de comparer des recherches avec différents systèmes de score.

e-value : correspond à l'espérance en probabilité, de trouver dans la base de donnée, un alignement avec un score supérieur ou égal à S. En pratique, plus il est faible, meilleure est la similarité (on ne regarde en général que les séquences ayant une *e-value* inférieure à 10^{-2})

[9] Dans la pratique on éxécute souvent sur les séquences retournées par Blast, un algorithme de SMITH & WATERMAN afin de rechercher le meilleur alignement local.

4.1.3 *"Grid Computing"*

Une grille de calcul consiste en un ensemble d'équipements informatiques connectés en réseau afin de mutualiser les ressources. Le concept de grille de calcul tend à s'approcher de celui d'un réseau électrique. En effet, ce dernier fournit, via une interface plus ou moins normalisée (la prise de courant) les ressources sollicitées. La complexité du réseau fournissant les services est masquée à l'utilisateur. Par ailleurs, la consommation d'un usager peut varier sans préavis. Ces trois points illustrent les principales caractéristiques d'une grille de calcul. Un cas d'utilisation simple est présenté en 4.3.1 à la page 29.

4.2 Problématique

Nous venons de montrer que les calculs nécessaires à la comparaison (et l'identification après analyse des résultats) de séquences sont grandement consommateurs en mémoire et temps de calcul (complexité en $O_{(n^2)}$ pour comparer deux séqences de taille n). Afin d'identifier la masse de séquences générées par le projet *TrichoEST*, il serait nécessaire d'investir dans l'achat de super-calculateurs extrêmement couteux. Une solution alternative à moindre coût consiste alors à mutualiser les ressources informatiques de l'entreprise grâce à la mise en place d'une grille de calcul, et développer une plateforme logicielle permettant son exploitation pour l'analyse et l'annotation de séquences.

4.3 Projets similaires, outils disponibles

La solution qui a été choisie n'est évidement pas unique, *Sun* a développé un système de grille de calcul *opensource*, et propose une solution à la parallelisation de Blast sur leur grille (7). Par ailleurs de nombreuses sociétés ont développé des solutions "*grille*" à ce problème comme Mesh-Technologies avec *OfficeGrid Blast*, Apple avec sa technologie *XGrid*. Le choix de InnerGrid a été effectué pour des raisons économiques, et afin de poursuivre un partenaria avec l'entreprise.

4.3.1 InnerGrid

InnerGrid est une plateforme logicielle développée et distribuée par la société GridSystem, qui permet, au sein d'un réseau hétérogène d'équipements informatiques, de bénéficier de la puissance de calcul disponible, et de manière plus générale, des ressources informatiques inutilisées.

Un exemple pratique d'utilisation d'une telle plateforme va nous permettre de mieux cerner l'intéret de cette pateforme. Considérons le thème de la conversion de formats vidéo. Ce sont en général des tâches nécécissitant des calculs relativement lourds, monopolisant ainsi un équipement dédié pendant une durée importante. Celle-ci peut être très largement réduite s'il est possible de morceler l'animation. Chaque fragment est ainsi attribué à un noeud de la grille, le serveur se chargeant alors d'orchestrer cet ensemble de calculateurs afin de mener à terme la tache globale.

Quelques mots sur la technologie mise en oeuvre dans cette plateforne : elle se fonde sur deux concepts centraux. Deux types de composants vont donc collaborer : les *Serveurs*, et les *Agents*[10].

[10] on parlera indistinctement d'agent, de client ou de noeud de la grille

- Le *serveur* est l'élément clé : il gère la distribution des tâches vers les agents. Il est également possible de travailler sur la même infrastructure avec un nombre plus important de serveurs, lorsque l'on se trouve dans un contexte où la tolérence aux pannes est critique[11].
- L'*agent* InnerGrid est la partie logiciel qui s'exécute chez le client, et qui permet à chaque noeud de la grille, de collaborer à la réalisation des tâches soumises au serveur. Ce logiciel s'exécute comme « tâche de fond »(ou encore « daemon », dans la langue de Shakespeare), et se charge ainsi de réaliser un ensemble d'actions nécessaires pour participer au processus.

Observons maintenant plus en détail comment la tâche est soumise puis réalisée sur la grille de calcul :

1. Le film original est découpé en petits fragments. Un cas intéressant est la fragmentation en autant de morceaux que de noeuds disponibles sur la grille.
2. Le serveur stocke sur son système de fichiers chacun des fragments, ainsi que les programmes exécutables utilisés pour effectuer la conversion. Dans le cadre (le plus courant) d'un réseau hétérogène, le serveur doit disposer des exécutables pour chaque plateforme afin de pouvoir fournir le programme adapté à chaque système d'exploitation.
3. Lorsque l'utilisateur lance une tâche de conversion, le serveur se charge de distribuer à chaque agent disposant des ressources nécessaires, et remplissant les conditions requises, un fragment du film ainsi que le programme exécutable, destiné à traiter ce dernier[12].
4. Chaque agent mémorise dans son système de fichier, le fragment de film ainsi que le programme de conversion. Il exécute alors sa tâche[13].
5. Lorsqu'un agent a terminé sa tâche, le fragment original est supprimé et le fragment converti est envoyé au serveur qui stocke alors le résultat sur son système de fichier(middleware).
6. Lorsque l'ensemble des micro-tâches de conversion est terminé, le serveur joint alors les fragments bout à bout pour construire le film converti.

4.3.2 BioJava

A l'instar de BioPerl, BioJava est un environnement de développement Java (6), permettant toutes sortes de traitements sur des données biologiques. Il inclue la création et manipulation de séquences, des parsers de fichiers, des accès au bases de données BioSQL. . .. Il se présente sur la forme d'une archive `jar` qui comporte toutes les bibliothèques de fonctions.

4.4 Plateforme logicielle : BlastOnAGrid

Cette section présentera les grandes lignes du fonctionnement de la plateforme. On se situe dans le cadre d'un nombre massif de données à traiter. Comme mentionné précédement, NBT coordine un projet internationnal de génomique fonctionnelle (TrichoEST. Les données issues du séquencage de l'organisme *Trichoderma* représentent maintenant quelques

[11] si par exemple un serveur tombe en panne, est victime d'une coupure de courant, ou se trouve deconnecté ou éteint par mégarde

[12] le serveur attribue, en fonction du système d'exploitation de l'agent, l'exécutable à joindre au « paquet »

[13] que l'on appelera aussi micro tâche si on se place au niveau du serveur, ayant une vue globale de la tache en cours (terminologie InnerGrid).

25000 séquences sur lesquelles il faut appliquer des comparaisons avec des bases de données gigantesques, qui permettent, par évaluation de leur similarité avec les séquences connues, de déterminer leur fonction. La figure 4.3 de la page 31 illustre le processus de traitement :

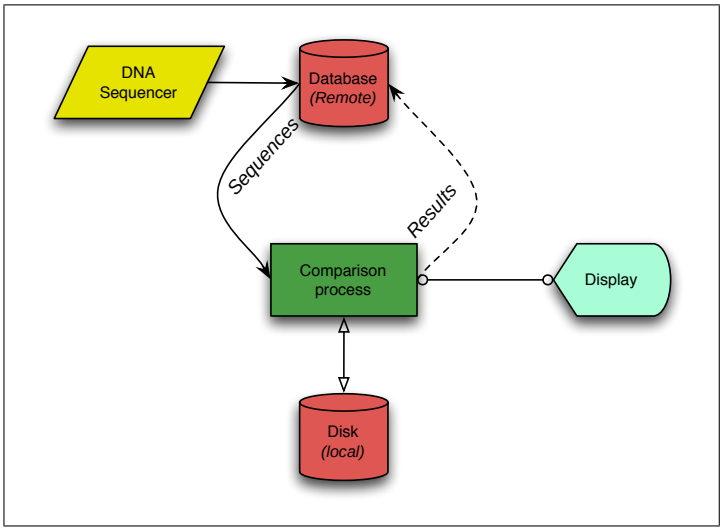

FIG. **4.3** – Les données issues du séquencage sont enregistrées dans une base de donnée, traitées localement, et les résultats sont eux aussi enregistré dans une base de donnée

Se pose alors un problème lié au temps de calcul nécessaire pour traiter (c'est à dire tenter d'identifier) ces données. Dans le cas de l'algorithme de *Smith & Waterman* qui n'a pas d'approche heuristique comme BLAST, la comparaison de deux séquences de taille n, m représente une complexité en $O_{(n*m)}$, ou plus généralement, dans le cas de n comparaisons de séquences de taille n, on atteint une complexité en $O_{(n^3)}$. L'achat de machine (super-calculateurs) dédiées représentant un investissement dépassant les ressources financières qui peuvent y être consacré, le pari est alors de faire collaborer toutes les ressources informatiques de l'entreprise par la mise en place d'une *grille de calcul*. Pour résumer, l'objectif de cette plateforme logicielle est de comparer de manière répartie les séquences biologiques, afin de pouvoir les identifier. On verra par aileurs que le *design* de cette architecture va permettre de développer rapidement une famille de logiciels fonctionnant sur le principe de la figure 4.3 en ne changeant que l'algorithme de traitement des données.

Les méthodes utilisées pour la réalisation de cette plateforme touchent tant à l'analyse et la comparaison des séquences biologiques, qu'à la gestion et à l'administration de bases de données, le "Grid Computing" et la modélisation et programmation orientée objet. Quant aux technologies, et langages utilisés, ils vont de l'API de INNERGRID NITYA fournie par *Gridsystem* à l'exploitation de Blast, Java, JDBC, MySQL, BioJava, Swing, UML, et la "galaxie" XML.

En reconsidérant le schéma de la figure 4.3 il est possible de donner plus de précisions sur l'infrastructure et les traitements - qui feront l'objet de la suite de cette section - qui s'organisent naturellement en 3 parties :

Pré-Traitements : Les séquences dont la fonction est inconnue sont stoquées dans une base de donnée et seules les séquences interessantes sont écrites localement sur le disque.

Traitements : Ces séquences sont comparées à la totalité d'une base de données regroupant des séquences connues avec le logiciel Blast.

Post-Traitements : Les résultats rendu par Blast sont alors "parsés" afin d'extraire les données interressantes, et ces dernières sont alors stoquées dans la base de données.

4.4.1 Pré-traitement des données

Parmi la masse de séquences à traiter, il s'agit de rappatrier celles qui nous intéressent. Elles proviennent d'un système de gestion de de suivi d'échantillons[14] qui permet, entre autre, leur stockage dans une table d'une base de donnée MySQL.

Afin de rendre BlastOnAGrid *ouvert* et facilement déployable au sein d'autres infrastructures informatiques, trois objets de configuration ont été créés. Ils permettent de mémoriser les paramètres de connection à la base de données, l'origine des données (nom de la base de données, de la table), la destination des résultats, et la configuration de la grille de calcul (adresse IP du serveur, port de communication utlilisé, groupe de machines auquel on attribue la tâche ...etc ...). Ces trois objets sont illustré par le diagramme statique de leur classes correspondantes à la page 52

Ce prétraitement est alors caractérisé par quatre étapes, et est réalisé en grande partie, par un objet de connection à la base de données. Il fournit des services d'ouverture et de fermeture de connection. Il permet aussi de réaliser les deux autres étapes qui sont, dans un premier temps, la création d'une table de descripteurs de séquences[15] - un descripteur étant caractérisé par l'identificateur de la séquence au niveau du SGS, et une description qui n'est rien d'autre que le nom de cette séquence - et par la suite, l'écriture locale des séquences au format FASTA. C'est un fichier texte dont la première ligne débute par le caractère > suivi de diverses informations sur cette séquence (les meta-données), et d'un retour à la ligne. La suite du fichier correspond à la séquence proprement dite, c'est à dire la suite des acides aminés ou des nucléotides.

Enfin, ce pré-traitement s'achève lors de l'envoi des séquences à la couche *middleware*, en constituant une archive au format Zip avec celles-ci. Cette opération est en effet nécessaire afin de ne pas surcharger le serveur de la grille de calcul. On a pu observer qu'au dessus de 1500 séquences envoyées, le serveur ne répond plus correctement aux requêtes. Il a donc fallu les envoyer sous la forme d'un fichier unique. Le choix du format Zip s'est imposé car l'API d'InnerGrid n'offrait que ce type de compression/décompression, au niveau du système de fichiers de la grille de calcul.

Ce pré-traitement peut donc se résumer ainsi : rapatriement local des séquences souhaitées, création de la table des descripteurs et constitution d'une archive Zip de ces séquences.

[14] SGS : Sistema de Gestión y Seguimiento de muestras

[15] qui nous servira tout au long du processus cf UML res package

4.4.2 Traitement des données

Le noyau de l'application a pour but d'envoyer, via la grille de calcul, une partition des données, à chacun des agents, en fonction de sa disponibilité et de ses ressources. Chaque agent traitera donc une fraction du travail global à effectuer. La figure 4.4 de la page 33 illustre un exemple d'infrasctructure pour ce type d'applications :

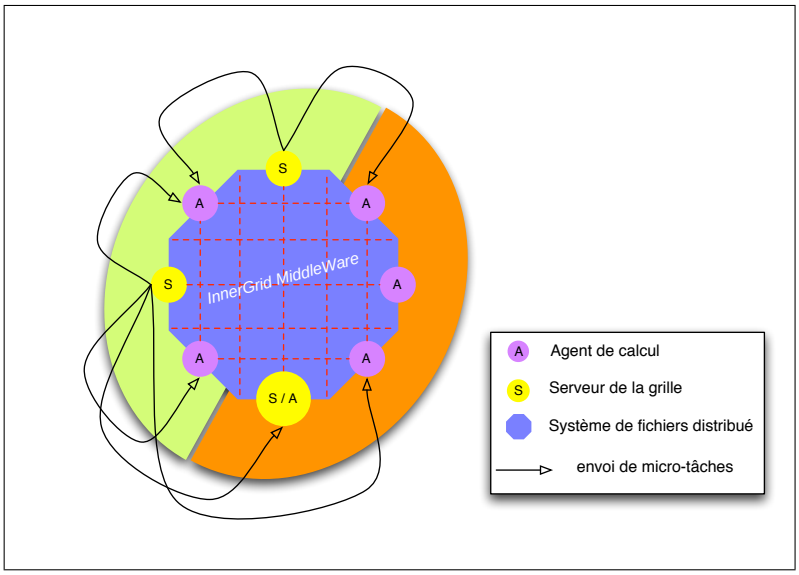

FIG. **4.4** – infrastructure possible lors de la mise en place d'une grille de calcul

Celle-ci est caractérisée par deux résaux locaux (LAN) au sein d'un réseau à grande échelle(WAN). Ce serait par exemple le cas de deux départements d'une même entreprise en deux lieux disctincts. Elle comporte deux serveurs (ou plus) afin de garantir une tolérence aux pannes, et de multiples agents. Il est également possible, dans le contexte d'une machine multi-processeurs, d'associer à celle-ci un serveur et un agent.

La définition du problème au niveau *middleware* s'effectue par deux concepts propres à INNERGRID : le *module* et la *tâche*.

Module : Par analogie aux domaines culinaires, il est constitué des ingrédients, à large échelle ; j'entend par là que l'on considère également les instruments de cuisine et les divers appareils et récipients. Les ingrédients à proprement parler seraient les séquences nucléiques (ou protéiques), les instruments de cuisine seraient quant à eux les éxécutables du programme BLAST pour chaque type d'agent, ainsi que la matrice des scores. Les détails de la configuration d'un module étant propres à INNERGRID, seules les idées sous-jacentes sont évoquées.

Tâche : En poursuivant cette analogie, une tâche correspond à la recette de cuisine, en d'autres termes, comment le travail à effectuer doit être divisé[16], le serveur affectant ainsi à l'ensemble des agents, une partition de la tâche. Dans notre contexte ce partitionnement est trivial puisqu'il consiste à envoyer à chaque agent, une séquence et un éxécutable adapté à l'architecture cible[17].

La figure 4.5 résume la constitution d'un module et d'une tâche pour une grille de calcul INNERGRID :

[16] dans la terminologie InnerGrid, comment doit-on passer d'une tâche à un ensemble de micro-tâches
[17] le résau informatique de NBT étant constitué de machines Linux, Mac et Windows

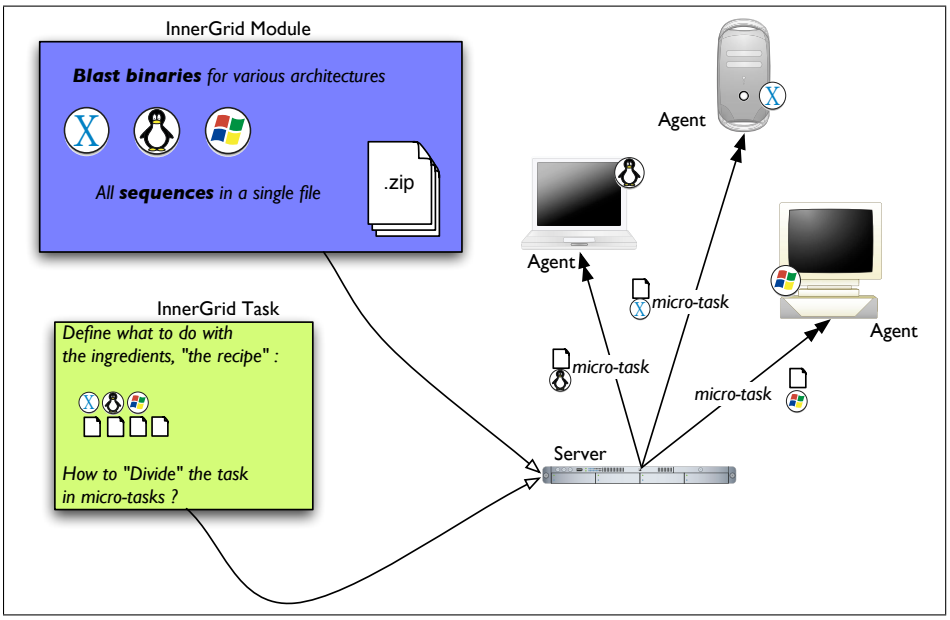

FIG. **4.5** – définition d'un travail à effectuer par la grille

4.4.3 Post-traitement des données

Blast produit deux types de résultats : une sortie directe au format texte, et une sortie balisée au format XML. Le problème revient alors à extraire les informations relatives aux séquences les plus proches permettant d'identifier celles sur lesquelles on travaille. Il s'agit donc de réaliser le *parsing* des résultas de Blast, et d'effectuer par la même occasion, après avoir créé une hierarchie d'objets abstraits correspondants aux résultats, l'actualisation de la base de donnée, avec ces objets.

Comme les calculs sont relativements longs, il n'est évidemment pas souhaitable que le logiciel se bloque dans l'attente de leur terminaison. La gestion des processus légers "*threads*" étant relativement facilitée par le langage Java, le module réalisant l'interface entre cette application et la grille de calcul a été développé de telle sorte que lorsqu'un résultat est « *prêt* » , il est automatiquement téléchargé et traité de manière indépendante. Ceci s'est avéré nécessaire du fait de l'hétérogénéité des machines, et de la variablilité de la taille des séquences à analyser. Les temps de calculs des micro-tâches étant donc très variables, les résultats apparaissent sur le système de fichier distribué, de manière sporadique. Une solution simple consiste à réaliser un *thread* se bloquant dans l'état « *en attente* » qui périodiquement (toutes les dix secondes par exemple) éxamine sur le disque local, la présence d'un résultat provenant de la grille. Dans ce cas, un autre *thread* se chargeant lui de "parser" le résultat est instancié et démarré. Cette solution, fondée sur l'attente active, est intéressante dans le cas d'un traitement massif des séquences, car les résultat sont envoyés à la base de donnée, de manière progressive.

En ce qui concerne le *parsing* proprement dit, il a été choisi d'utiliser la bibliothèque BIOJAVA qui permet de traiter des données biologiques. Parmi la masse des services qu'elle peut fournir, elle contient déjà un parser permettant de réaliser l'extraction d'informations d'un rapport de résultats Blast. Un travail préalable à la mise en place d'un parser a été, de créer un modèle des résultats fournis par les programmes de type Blast (ou *Blast-like*, comme WU-Blast, HMMER, DBA). Ce travail s'est fortement appuyé sur l'analyse de la grammaire d'un tel modèle sur laquelle est fondée le parser de BIOJAVA, la classe `BlastLikeSAXParser`. Cette grammaire est une DTD XML, disponible dans le répertoire CVS de BioJava (6) dans

le répertoire `[biojava]/biojava-live/dtd`. De cette DTD ont été sélectionné les données[18] qui se sont avérées dans notre contexte les plus intéressantes. `BlastLikeSAXParser` se charge de parser le rapport Blast au format texte comme s'il s'agissiat d'un fichier XML écrit suivant la DTD de BioJava. Il ne reste donc plus qu'à écrire la partie évènementielle[19] de ce parser, comme si on travaillait avec un parser SAX, le gestionnaire de contenu "*ContentHandler*". Il se charge de réagir aux évènements XML. Trois évènements sont essentiels pour l'implémentation de cette interface :

startElement : il notifie une balise ouvrante d'un élement XML.

endElement : indique une balise fermante d'un élement.

charcters : les éléments XML ne sont pas formés que de balises et d'attributs, ils peuvent aussi contenir du texte entre une balise ouvrante et une fermante. Cet évènement est levé dans ce cas.

Lorsque le parser réagit à ces évènements, les instructions de création des objets résultats sont éxécutées, et lorsqu'un tel objet est terminé, il éxécute sa méthode `sendMe()` qui réalise son enregistrement auprès de la base de donnée.

J'ai choisi de ne pas utiliser les diagrammes de séquence de la syntaxe UML, afin de communiquer plus facilement avec mes collègues n'ayant pas une culture "informatique", et "modélisation objet". Le shéma de la figure suivante est une représentation qui peut sembler curieuse, mais résumant le fonctionnement général de l'application.Cette illustration tente de représenter les deux dimensions du système : spatiale et temporelle. Les flèches en traits pleins, représenteront les séquences, celles en pointillés représnteront les résultats issus de BLAST.

[18] Un modèle UML ayant servi au codage de cette partie est visible en annexe a la page 52.

[19] des balises ouvrantes ou fermantes peuvent être considérées comme des "évènements XML" générés par le parser

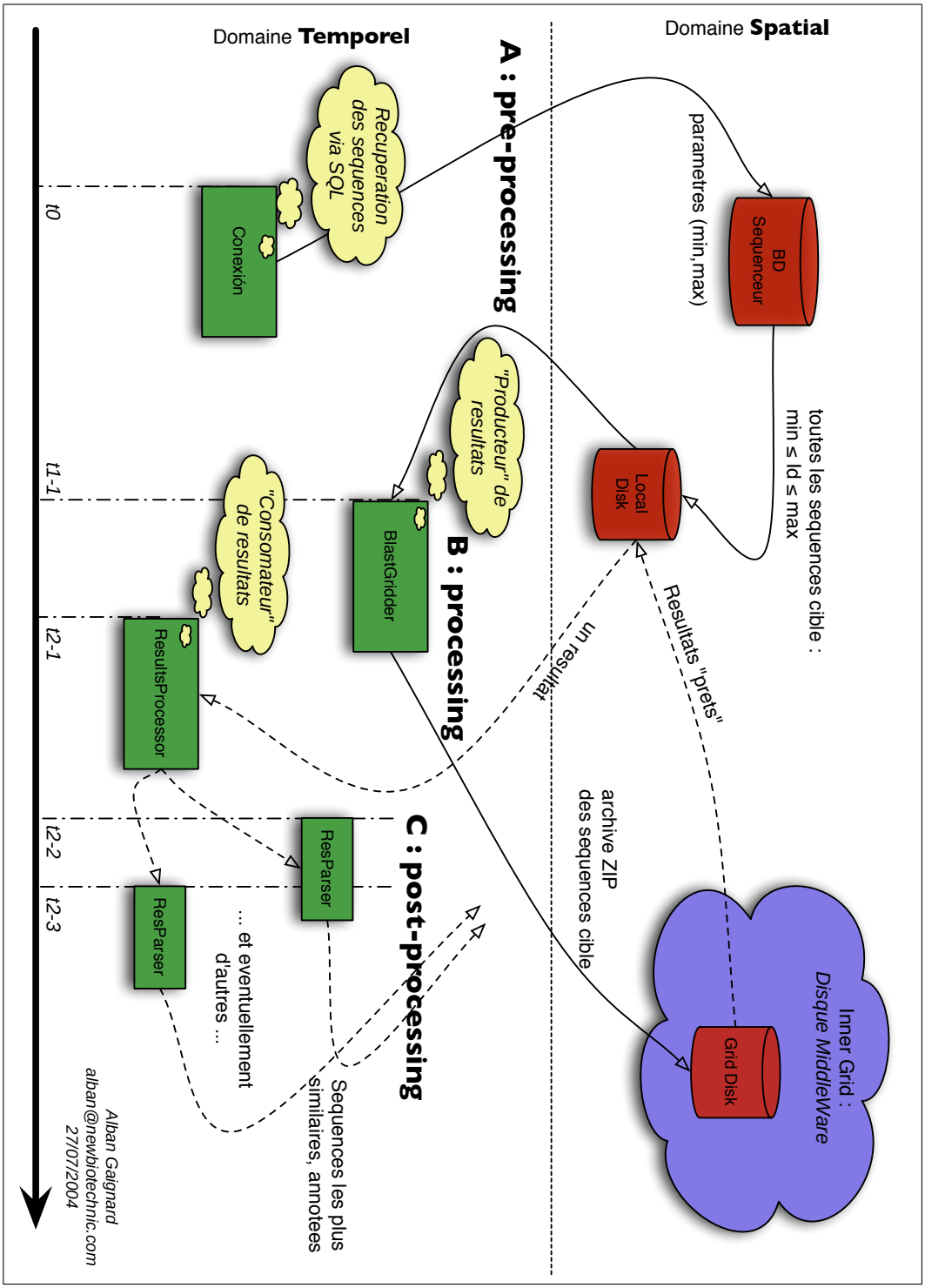

FIG. 4.6 – séquence d'exécution des modules de l'application ; sources et destinations des flux de données

4.4.4 Choix de conception, modélisation

Ce paragraphe justifiera dans un premier temps le choix du langage de programmation et précisera par la suite certains points ayant trait à la modélisation objet du système.

Une telle plateforme peut être vue de manière minimaliste comme un simple *pipeline* de traitement de données. Il aurait pu en effet se présenter sous cette forme : aquisition d'un ensemble de séquences provenant d'une base de données, envoi de celles-ci à la grille de calcul, rapatriement des rèsultats, puis enfin, envoi de ces derniers à une base de données. Le choix du langage de programmation a été contraint par le kit de développement InnerGrid *SDK*. Il nous contraint en effet à choisir parmi deux langages dits *Orientés Objet*, C++ et Java, et un langage de script fameux dans la communauté scientifique pour ses nombreux *packages* de traitement de données biologique, et de chaînes de caractères : Perl. Voici un bref aperçu des technologies à mettre en oeuvre dans ce type de systèmes : on doit être capable d'accèder à une base de donnée, y effectuer des requêtes et des mises à jour ; manipuler des structures de données complexes comme des rapports de comparaison de séquences génomiques ; traiter des flux XML ; éventuellement proposer une interface via le Web, par des pages ou une applet, ou une interface graphique indépendante. Par ailleurs, les traitements des séquences par chaque agent sont effectués de manière indépendante. Les résultats apparaissent alors de manière asynchrone. On doit donc être également capable de gérer et synchroniser des processus paralléles (comme ceux traitants les résultats). Le *panel* est donc relativement étendu ... Voici les raisons qui m'ont fait opter pour Java :

- **portabilité :** le parc des machines de NBT est relativement hétérogène.
- **richesse de fonctionnalités :** permet de réaliser simplement des accès à des bases de données par des *ponts* JDBC ; présence de nombreuses bibliothèques de fonctions pour le traitement de flux XML.
- **gestion des threads :** facilité de synchroniser les *thread* et verrouiller(par exclusion mutuelle) les données partagées, par exemple la table de descripteurs de séquences utilisée tout au long du processus.

Le modèle figurant en annexe C.2, a permi d'envisager le développement d'autres type de traitements de ces mêmes séquences, comme mentionné précédement. Par ailleurs, pour rendre la manipulation des séquences plus facile et afin de limiter les requêtes à la base de données [20], j'ai choisi de mémoriser au cours de l'exécution de l'application, des descripteurs constitués d'un identificateur, et du nom de la séquence. Seul l'identificateur, ou la description de la séquence aurait suffit, il semble donc possible d'optimiser l'usage de la mémoire en ne mémorisant que le nom de la séquence, ou en ne mémorisant que son identificateur.

4.4.5 Interface Homme-Machine

Bien que fonctionnelle, cette application n'est toujours pas exploitable par des utilisateur non-programmeurs. En effet, il s'agit d'offrir à l'utilisateur final, plusieurs choses :

- indiquer au logiciel la source des séquences et la destination des résultats.
- indiquer òu se situe le serveur de gestion de la grille et comment ỳ accéder.
- choisir le type de traitement à effectuer sur les séquences
- contrôler le démarrage, la pause, et la suppression d'une tâche éxécutée par la grille de calcul et en visualiser la progression

J'ai réalisé à l'aide de Swing, une interface graphique se présentant sous la forme d'une fenêtre unique avec différents onglets, permettant de configurer les sources et destinations

[20] il y a des phases de lectures et d'écritures, on nécessite en plus de la séquence, son nom et son identificateur.

des données, la grille de calcul, la sélection des séquences et le choix du traitement à effectuer.

4.4.6 Diffucités techniques rencontrées

Certaines difficultés ont été éprouvées à l'utilisation du parser `BlastLikeSAXParser` fourni par la bibliothèque BioJava. Je pensais en effet que le fichier d'entrée de celui-ci etait la sortie XML de Blast et non la sortie "texte". L'examen des archives de la liste de diffusion mail de BioJava m'a été d'une grande aide. Une autre difficulté a été liée à la prise en main de la grille de calcul InnerGrid et de la constitution "à la main" du module Blast à distribuer. Une autre difficulté est liée au déploiement de cette application, qui doit être simple et comporter toutes les librairies nécessaires, en limitant les conflits avec les différentes versions de la *Java Virtual Machine*. Ce qui a nécessité un grand investissement, a été de faire comprendre que la modélisation UML, n'avait pas pour principal objectif de documenter l'application, sinon, de se donner des bases communes pour réfléchir à sa conception.

4.4.7 Evaluation des performances

Auparavant, les séquence étaient comparées via l'inteface web du NCBI[21] et nécessitaient entre 3 et 5 minutes de calcul, sans compter la mise en file d'attente lorsque le serveur est sollicité par de nombreuses requêtes. Du fait du caractère non-dédié des machines de la grille, chaque séquence est traitée plus lentement, mais le gain est notoire si on considère la parallèlisation d'un grand nombre de traitements. Des tests ont été réalisés en lançant la comparaison de 2000 séquence sur la grille et le *speedup* atteint était de l'ordre de 9. On va donc 9 fois plus vite que si la comparaison était lançée sur un seul équipement. Cette rapidité est théoriquement proportionnelle au nombre de noeuds de la grille.

4.5 Mise en perspective

J'avoue avoir été parfois un peu déconcerté quant à la difficulté de faire prendre conscience de l'importance de la modélisation objet de ce projet, mais je suis satisfait d'avoir pu me faire comprendre, et avoir fait projeter le développement d'une famille de logiciels sur ce même modèle, en ne changeant que le noyau de traitement et le parser de résultats. Ces projets ont d'ailleurs été approuvés au sein de l'entreprise car un nouveau stagiaire a été recruté pendant mon dernier moi de stage afin de poursuivre le travail dans ce sens, c'est à dire développer un autre module de traitement des séquences, accompagné du parser de ses résultats (Interproscan). Par ailleurs, la commercialisation de ce type d'application est envisagée lorsque sera mise en place une base de donnée de démonstration.

[21] *National Center of Biotechnology Information*

Chapitre 5

Prise de recul et conclusions

5.1 Organisation

Ce séjour au sein de l'entreprise NEWBIOTECHNIC, constituant pour moi une découverte sérieuse du monde de l'entreprise, s'est déroulé principalement en trois parties, chacune faisant l'objet d'un chapitre de ce présent document. Le premier mois et demi a été consacré à la découverte de l'entreprise, de ses outils et de ses infrasctructures informatiques, et à l'élaboration du système de quantification de croissance de racines par analyse d'images. Le mois et demi suivant a quant à lui été dédié à la maintenance et l'évolution de l'application Web (GIO). Les 3 derniers mois ont été consacrés à la conception et la mise en oeuvre du système de comparaison répartie de séquences génomiques.

J'ai effectué mon travail suivant les horaires des autres employés avec une certaine flexibilité : huit heures de travail journalier, généralement de 9h à 14h et de 16h à 19h, avec un changement d'horaire durant l'été, plus adapté aux conditions météorologiques à cette période de l'année, de 8h à 15h.

Le poste de travail auquel j'ai été affecté s'est situé dans un espace ouvert, relativement petit[1] , où travaillent les biologistes (en dehors du laboratoire), le directeur commercial et marketing, les personnes chargées des thèmes bioinformatiques, et où se situe également le secrétariat.

5.2 Difficultés d'ordre général rencontrées

Sur des aspects plutôt logistiques, j'ai éprouvé les mêmes difficultés que la majeure partie des employés travaillant dans cet espace ouvert, liées principalement au niveau sonore lors des communications téléphoniques nécessaires au travail de chacun.

Concernant le thème de l'analyse d'image, la principale difficulté a été plutôt liée à une adaptation de langage à effectuer. Il a fallu en effet trouver un langage commun permettant de manipuler les notions de traitement d'images sans considérations techniques, trouver également la terminologie (souvent aidée par l'élaboration de shéma et de dessins) adaptée, afin d'expliquer pourquoi des erreurs de mesures étaient introduites par l'algorithme d'imageJ et pourquoi il n'a pas été développé de solution alternative.

Les difficultés rencontrées lors du travail sur l'application Web ont principalement été induites par le stade non définitif de son élaboration, et par l'inexistence de documents de conception. Le temps passé à l'élaboration d'une documentation a été parfois déconcertant.

[1] L'entreprise étant en pleine expansion, son déménagement dans d'autres locaux est en cours

J'ai par ailleurs dû insister grandement sur la nécessité de l'élaboration d'un modèle UML de la plateforme d'analyse de séquences, et montrer que c'était un élément clé pour la maintenabilité de l'application, et une aide précieuse à la conception de celle-ci. Ce modèle s'est avéré d'ailleurs utile pour la simplification de l'attribution des tâches dans le contexte d'un travail en équipe, notamment à la fin de mon séjour lorsqu'un nouveau stagiaire a été embauché afin d'étendre la plateforme par le développement d'un nouveau module.

5.3 Travail réalisé

Sur un plan général, plusieurs tâches d'administration de systèmes m'ont été attribuées au cours de ce séjour, comme par exemple la migration vers linux d'un certain nombre de machines de l'entreprise, et la formation rapide de mes collègues à l'utilisation de la suite bureautique *openoffice* et à l'utilisation d'un client mail différent de Outlook. J'ai eu aussi a résoudre des problèmes liés au réseau local et à la coopération entre les systèmes d'exploitation Linux, Mac OS X, et diverses version de Windows.

Une des réalisations techniques est une application Java fondée sur ImageJ réalisant la segmentation d'une photo de plusieurs racines de plantes afin de mesurer leur croissance. Cette application est effective et donne des résultats intéressant car le gain de temps dans l'analyse de ces photos est notoire.

J'ai réalisé des documents permettant de maintenir l'application Web (GIO) sous la forme de deux graphes, l'un étant axé *application*, l'autre étant axé *communication* de données entre les pages. J'ai par ailleurs fait évoluer celle-ci en ajoutant des fonctionnalités d'importation de données météorologiques, et en testant la possibilité de générer automatiquement les résultats, avec une présentation adaptée au format papier.

Le travail le plus important que j'ai réalisé durant la moitié de ce séjour consiste en la conception et au développement d'une plateforme de comparaison répartie de séquences génomiques fondée sur la programmation de la grille de calcul InnerGrid. Elle permet l'éxécution en parallèle du programme Blast sur une base de données locale de séquences génomiques. Le gain de temps, avec les machines disponibles dans l'entreprise, est allé lors des tests jusqu'à un facteur 9. Le nombre de séquences à analyser étant très important, ceci représente un gain de temps considérable.

Dans ces trois domaines, j'ai proposé par ailleurs activement mon avis et mes réflections sur les améliorations à apporter et sur les compromis effectués.

5.4 Conclusion

Ce séjour en entreprise m'a permis d'apporter à Newbiotechnic des solutions techniques et méthodologiques à certaines nécessités exprimées, de manière formelle ou informelle, comme par exemple la migration vers des outils libres de droits, les méthodes de développement *orientées objet*, et des solutions effectives pour les trois principaux thèmes de travail.

Newbiotechnic a pu m'apporter un certain nombre de connaissance générales, dans l'application des sciences informatiques à des problèmes scientifiques concrets, et à effectuer des compromis entre temps de développement et efficacité de la solution proposée. Plus précisément, j'ai appris auprès de personnes particulièrement confirmées beaucoup de concepts et techniques (incluant les notions de biologie sous-jacentes) dans l'analyse fonctionnelle de séquences génomiques, et plus généralement en bioinformatique. N'étant pas spécialiste en analyse d'image j'ai eu la possiblité de me documenter sur ces thèmes et d'acquérir ainsi de nouvelles connaissances, puis de les mettre en pratique dans le cadre de ce séjour. Les

travaux effectués sur l'application Web m'ont permis de développer une certaine rigueur dans l'élaboration de sa cartographie.

Cette collaboration avec l'entreprise m'a apporté énormément, et s'est déroulé dans un climat d'écoute, d'échanges, de bonne humeur, propice à un travail efficace. Elle a débouché sur une proposition d'emploi à laquelle j'ai du particulièrement réfléchir. Je me suis finalement décidé sur une autre offre à laquelle j'ai répondu et qui a été positivement éxaminée. Nous allons donc rester en contact afin d'éventuellement collaborer à nouveau, si je deviens disponible et décide de me consacrer pleinement à la bioinformatique. Sur un plan plus personnel, j'ai particulièrement apprécié ce long séjour en Andalousie, qui a été riche en échanges culturels, et nouvelles amitiés.

Annexe A

Mesure de croissance de Racines via ImageJ

A.1 Fonctionnement de l'application de mesure

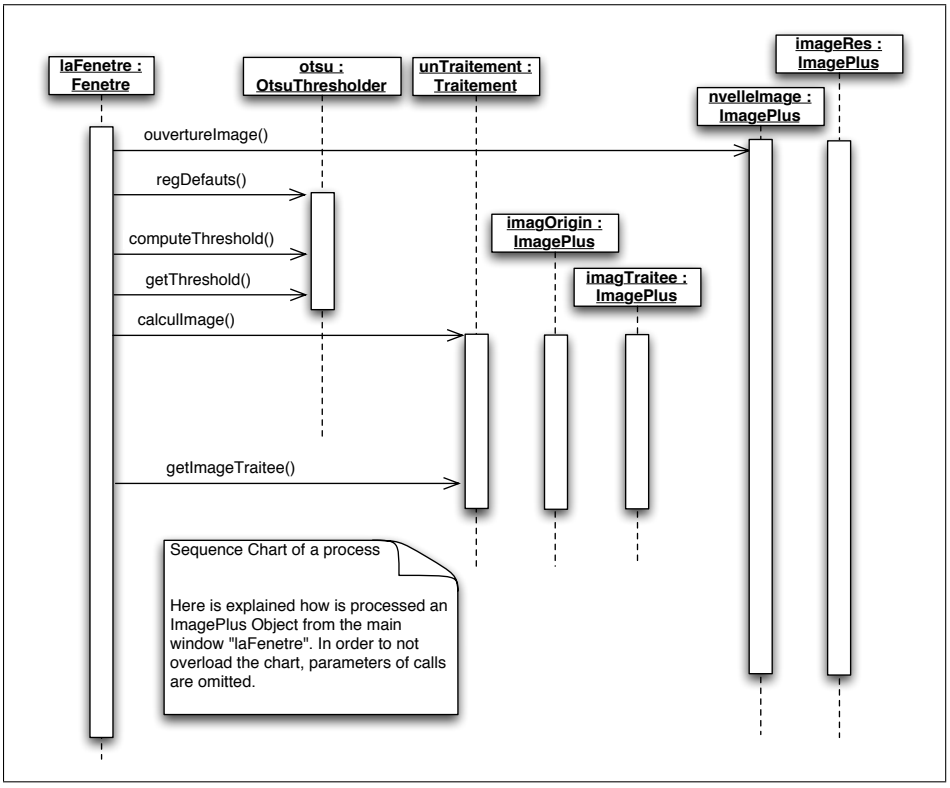

FIG. **A.1** – Diagramme de séquence UML lors d'un traitement d'une image

A.2 Document diffusé au sein d'NBT, au département "phytopathologie"

Roots area measurement through ImageJ

pre-release document, thanks to send me back suggestions :

`alban@newbiotechnic.com`

5th April 2004

This document describes how you can measure automatically roots, with the free software ImageJ (`http://rsb.info.nih.gov/ij/`*,* Java *is required). The steps are exposed below, in the first section. A second section will describe automatic processing through an* ImageJ *plug-in*

A.3 Manual processing

A.3.1 Image to process : the photo of roots

First of all, we need to open the software, in Windows or MacOS, a simple double-click on `IJ.jar` will works well, in Linux or Unix-like, we need to execute the script `run` in a shell with `./run`. Then, using the standard menu, we open the photo (`File` → `Open...`) :

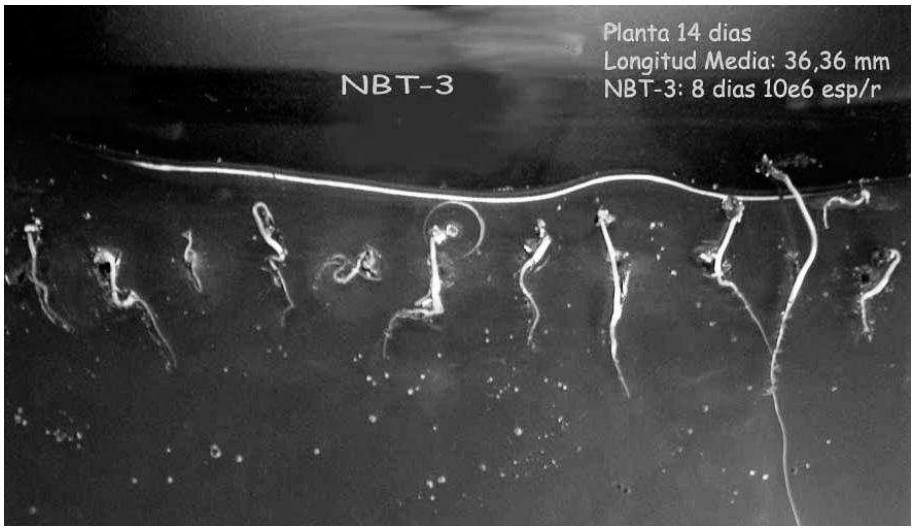

Figure **A.2** : Roots with unknown area

A.3.2 Pre-processing

This step consists in converting nicely the color photo to black and white. We need to transform is type to Gray-8-bits. It's done by hitting the menu `Image` → `Type` → `8-bits`. Eventually we need to adjust the contrast by the

`Image` → `Adjust` → `Brightness/Contrast...` menu.
The critical point consist in setting nicely the parameters of the Threshold filter : `Image` → `Adjust` → `Threshold...`

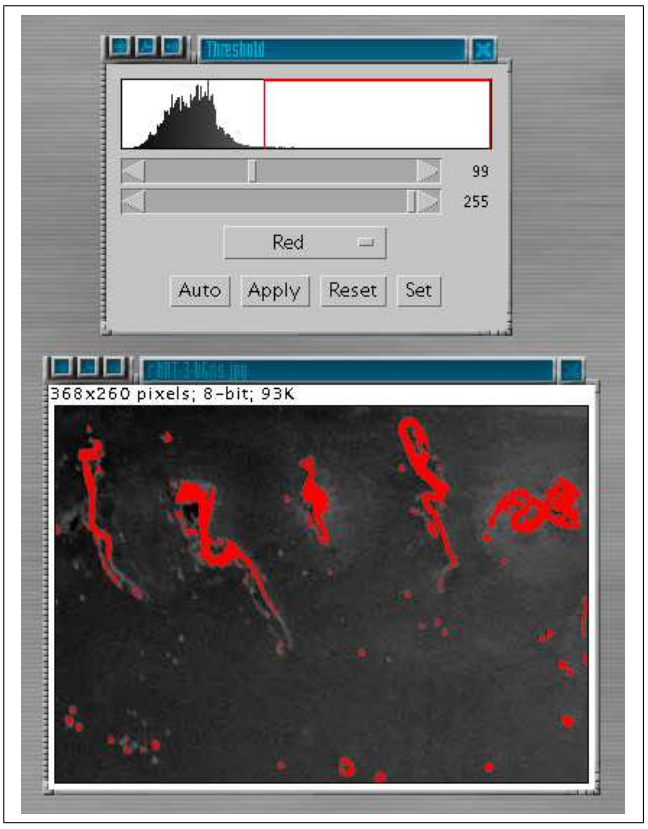

Figure **A.3** : Conversion of a gray image to a binary image

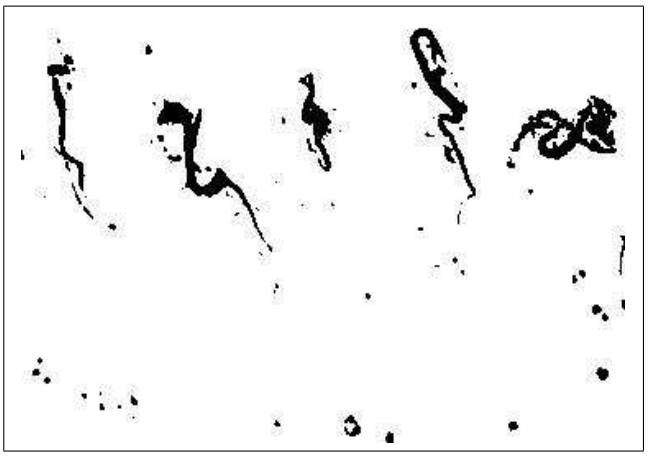

Figure **A.4** : Image used to determine the area of roots

A.3.3 Analysis of particle : counting and measurement

The first step consists in setting up the measurement (`Analyze` → `Set Measurement...`) as in the following picture :

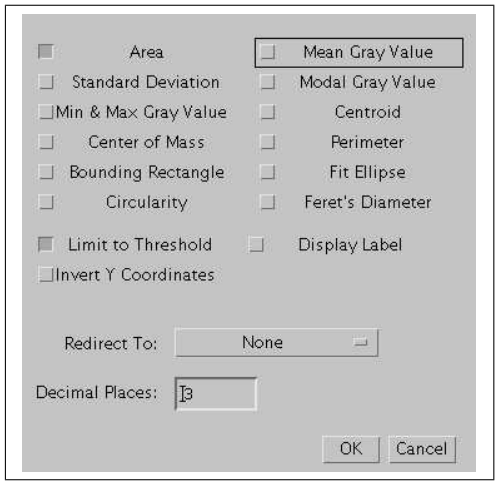

Figure **A.5** : Parameters of measurement

Then, the filter particle analysis (`Analyze` → `Analyze Particles...`) can be applied :

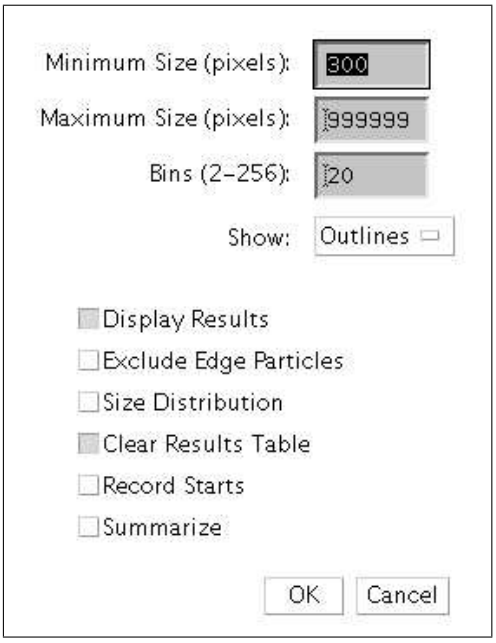

Figure **A.6** : Parameters of the analysis

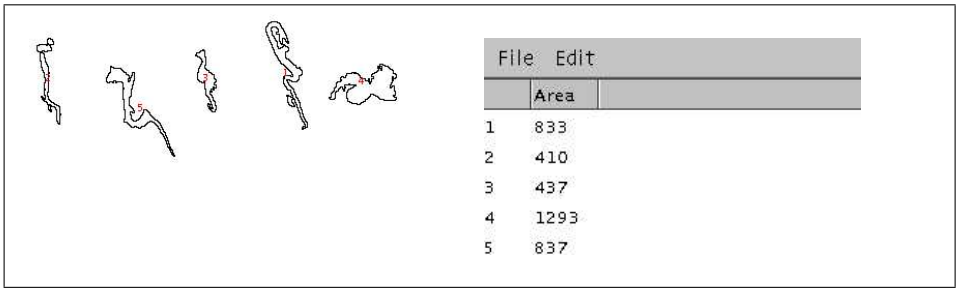

Figure **A.7** : Results of the analysis

A.3.4 Last minute ...

An other method to pre-process the source image, consists in first, applying a high-pass filter. It can be done by executing a convolution filter with kernel :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Then, you should run the filter `Substract Background...` of the `Process` menu and choose the black or white background. Now you might be able to threshold the image more easily. The ultimate thing is the parametrisation of binary filters : `Process` → `Binary` → `Options...`, by setting the number of iterations to 1, and choosing the color of the background (usually, it's white). Then, in the same menu, you can operate the morphologic operation "close", in order to "reconnect" particules and smooth shapes.

A.4 Automatic processing

A.4.1 How To ...

It's supposed that an 8-bits gray image is opened (see the first section of this document), and that a region of interest is selected (the first tool of the ImageJ's toolbar). In order to enhance contrasts, you can adjust it, as in the pre-processing section. Then, you execute the plugin in the `Plugins` menu.
In the case of a system failure ... installing it from scratch consists in installing the software ImageJ anywhere on the harddrive, and executing the plugin manually by hitting the menu `Plugins` → `Compile and run ...` and browsing as the location of the file `CUANTIFICACION_DE_RAICES.java`.

A.4.2 Summary of the process

1. determine the ROI (Region Of Interest)
2. create a new ImageProcessor, and then a new image, copy of the ROI
3. process a HighPass filter to let imageJ determine efficiently the shapes
4. set up the Threshold filter by the parameters min and max
5. finally, tell the plugin Particle Analysis with the flags
   - `SHOW_RESULTS`
   - `SHOW_PROGRESS`
   - `SHOW_OUTLINES`
   - `CLEAR_WORKSHEET`

   with also the second parameters as
   - `AREA`
   - `LIMIT`

   and the others parameters as a new ResultTable, the min size of a particle (300), and the max size (999999).

A.4.3 Things that are not yet implemented

- the high pass filter is up-to-date but not executed yet in the process.
- the morphologic "close" operation might be a solution to connect artificially particules disconnected by the Threshold filter.

A.4.4 *To Do . . .*

- execute a High-Pass filter (finding a good kernel of convolution).
- determining the min and the max of the setThreshold method.
- eventually inverse the image.
- execute a "closure" on the binary image.
- designing a Graphical User Interface ... hiding the ImageJ software.
- developping a version that can be used, as an applet, or as a command-line application.

Annexe B

Gestion Intégrée de l'Oliveraie

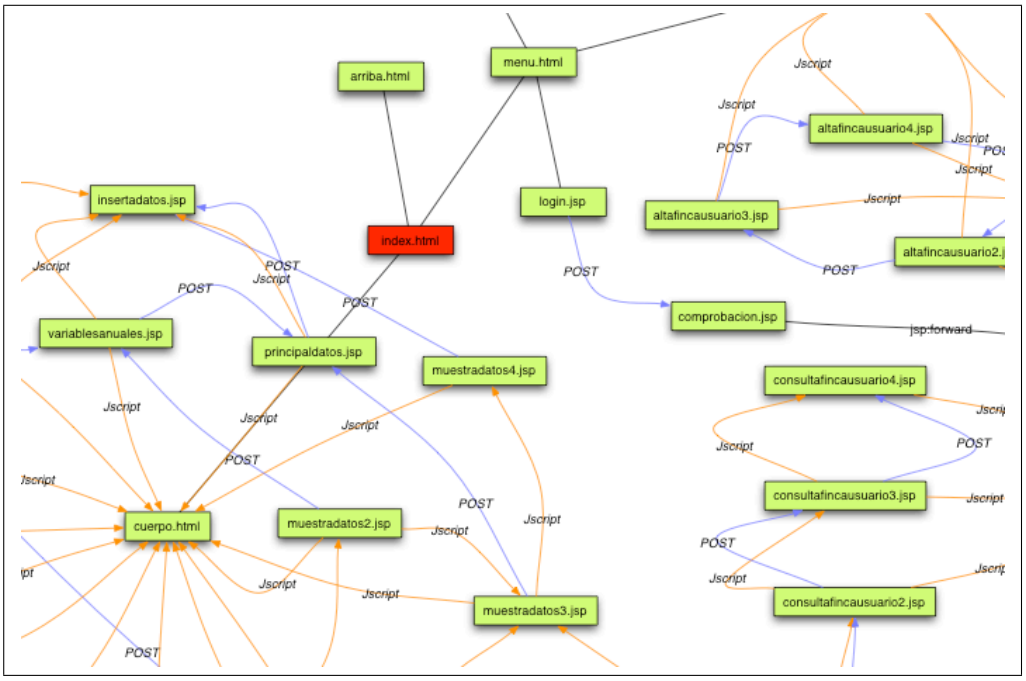

FIG. **B.1** – Extrait de la cartographie de l'application Web axée *communication*

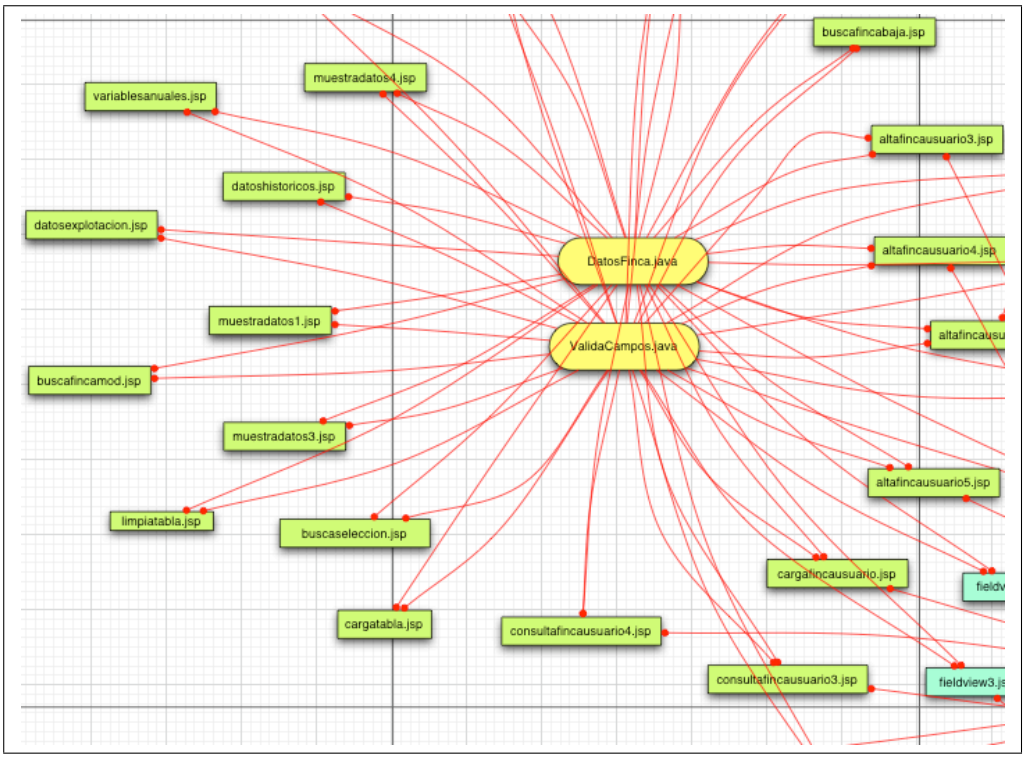

FIG. **B.2** – Extrait de la cartographie de l'application Web axée *application*

Annexe C

BlastOnAGrid

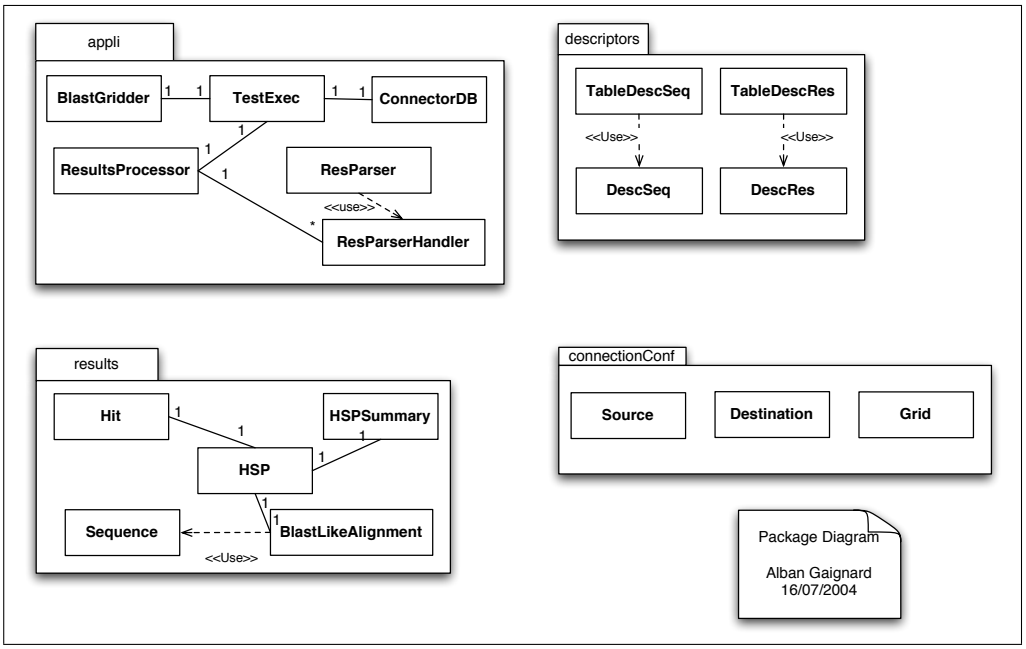

FIG. **C.1** – ensemble des packages du programme BlastOnAGrid

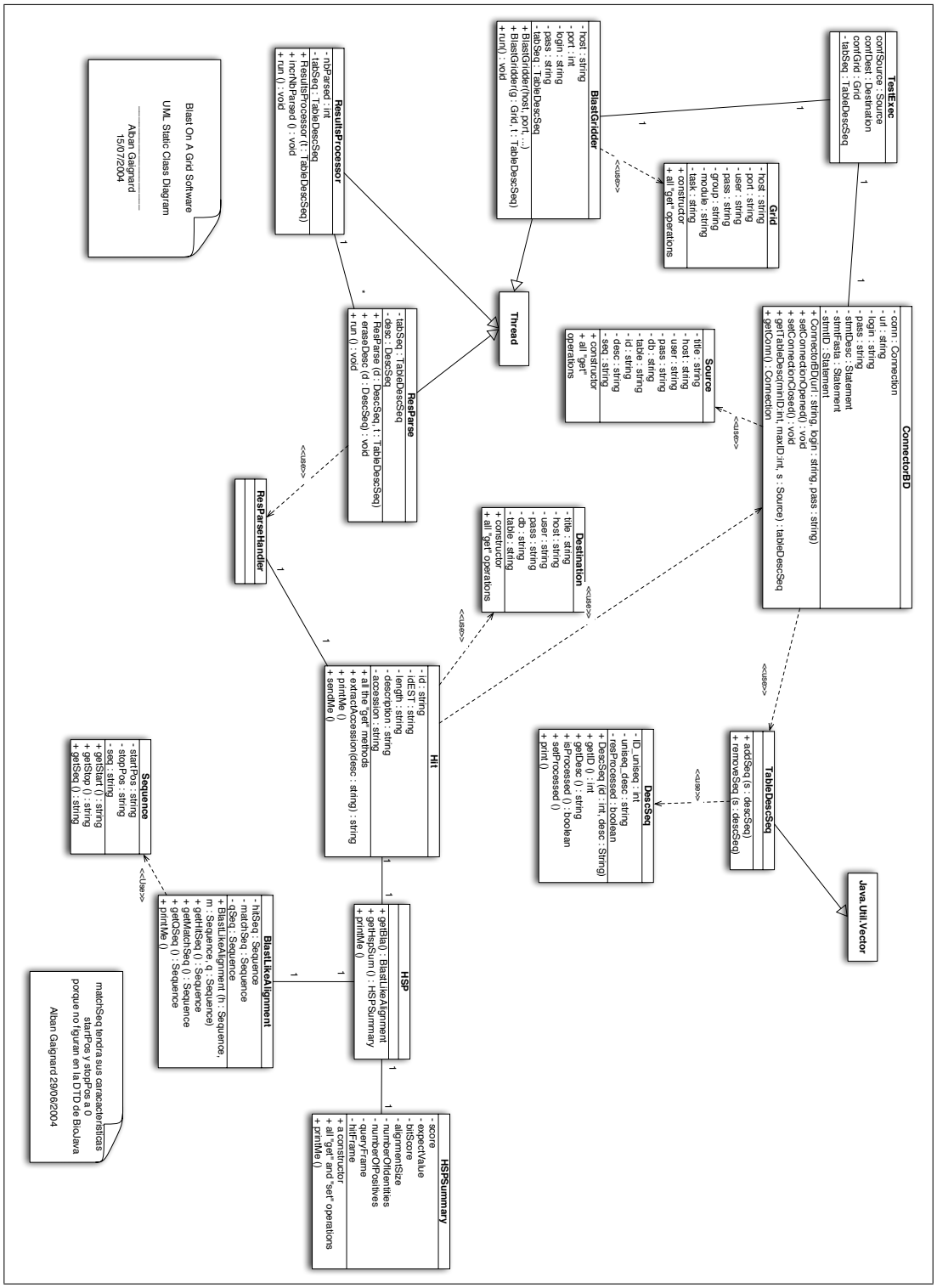

FIG. **C.2** – modèle statique de BlastOnAGrid

Bibliographie

[1] Ron Shamir, Dana Torok & Adar Shtainhart. *Analysis of Gene Expression Data*, mars 2002.

[2] Centre de Ressources InfoBioGen *Portail de Biologie et de Bioinformatique* http ://www.infobiogen.fr/services/deambulum/fr.

[3] Ian Korf, Mark Yandell & Joseph Bedell. *BLAST, An Essential Guide to the Basic Local Alignment Search Tool*, O'reilly, 2003.

[4] Stéphane Guyetant. *Evaluation d'heuristiques pour l'alignement de séquences génomiques.* Irisa, 2003.

[5] Rex A. Dwyer. *Genomic Perl, from bioinformatics basics to working code*, Cambridge University Press.

[6] Thomas Down, Michael Heuer, David Huen, Matthew Pocock & Mark Schreiber. http ://www.biojava.org. http ://cvs.biojava.org. http ://www3.sympatico.ca/foisys/bj_in_anger.

[7] Sun Microsystems, Bogdan Vasiliu. *Integrating BLAST with Sun ONE Grid Engine Software.* Juillet 2003. http ://developers.sun.com/solaris/articles/integrating_blast.html.

[8] Wellington S. Martins, Juan del Cuvillo, Wenwu Cui & Guang R.Gao. *Whole Genome Alignment using a Multithreaded Parallel implementation.*

[9] Aurélien Leroux. *Sampling : entrepôt de séquences biologiques construit sous forme d'agent logiciel et spécialisé dans la comparaison intensive de séquences.* Rapport de stage de DEA, Irisa & Inserm-Germ 435.

[10] F. Hernandez, N. Jacq & S. Nicoud. *DataGrid, Projet Européen de grille de calcul*, 2001.

[11] Choon Hoong Ding & Rajkumar Buyya *Guided Google : A Meta Search Engine and its Implementation using the Google Distributed Web Services.*

[12] Sébastien Meric. *Lecture d'un flux XML via SAX.* Octobre 2003. http ://smeric.developpez.com/java/cours/xml/sax/.

[13] Stéphane Bres, Pierre-Antoine Champin, Jean-Mathias Heraud, Vincent Herilier, Jean-Michel Jolion, Etienne Loupias. *Traitement d'images et Vision Artificielle* http ://telesun.insa-lyon.fr/~telesun/.

[14] Bruno Nazarian. *Cours d'imagerie numérique* http ://bnazarian.free.fr/Cours/Cours.htm

Table des figures

2.1 Du pilotage des fonctionnalités d'imageJ à la présentation des résultats 10
2.2 Images à traiter en début et fin d'élaboration du système de quantification de croissance . 13
2.3 plugin réalisant l'inversion d'une zone prédéterminée de l'image 14
2.4 suite des traitements appliqués à l'image à traiter 16
2.5 IHM de l'application de quantification de croissance de racines 17
2.6 dessins illustrant les défauts de l'algorithme 18

3.1 légende des graphes représentant l'application web 21

4.1 dogme central de la biologie moléculaire 25
4.2 processus de séquencage d'un *EST* 26
4.3 Les données issues du séquencage sont enregistrées dans une base de donnée, traitées localement, et les résultats sont eux aussi enregistré dans une base de donnée 31
4.4 infrastructure possible lors de la mise en place d'une grille de calcul 33
4.5 définition d'un travail à effectuer par la grille 34
4.6 séquence d'exécution des modules de l'application ; sources et destinations des flux de données 36

A.1 Diagramme de séquence UML lors d'un traitement d'une image 42
A.2 Roots with unknown area 43
A.3 Conversion of a gray image to a binary image 44
A.4 Image used to determine the area of roots 45
A.5 Parameters of measurement 45
A.6 Parameters of the analysis 46
A.7 Results of the analysis 46

B.1 Extrait de la cartographie de l'application Web axée *communication* 49
B.2 Extrait de la cartographie de l'application Web axée *application* 50

C.1 ensemble des packages du programme BlastOnAGrid 51
C.2 modèle statique de BlastOnAGrid 52

Liste des tableaux

4.1 Remplissage de la matrice de programmation dynamique avec l'algorithme de Needlman & Wunsch 27
4.2 Backtracking pour determiner l'alignement global 27
4.3 matrice de l'algorithme de Smith & Waterman 28